



UNIVERSITÀ DEGLI STUDI DELL'AQUILA  
DIPARTIMENTO DI  
INGEGNERIA E SCIENZE  
DELL'INFORMAZIONE E MATEMATICA



CORSO DI LAUREA IN  
INGEGNERIA INFORMATICA E AUTOMATICA

**Sviluppo di un sistema di stima dell'assetto per  
un'applicazione di localizzazione indoor**

**Relatore:**

*Prof. Luigi Pomante*

**Candidato:**

*Fabio Di Sabatino*

**Co-relatori:**

*Giacomo Valente, Marco Santic*

**Matricola:**

*246526*

---

ANNO ACCADEMICO 2017–2018

# Indice

<b>Introduzione</b>	<b>1</b>
<b>1 Contesto applicativo</b>	<b>4</b>
1.1 Context-aware computing . . . . .	4
1.2 Indoor Positioning System Service . . . . .	6
1.3 Stima della posizione . . . . .	8
1.3.1 Range based . . . . .	9
1.3.1.1 Received Signal Strength Indicator - RSSI . . . . .	9
1.3.1.2 Time Of Arrival measurements . . . . .	11
1.3.1.3 Time Difference Of Arrival . . . . .	12
1.3.2 Angle Based . . . . .	13
1.3.2.1 Angle of Arrival . . . . .	13
1.4 Tecniche di localizzazione . . . . .	15
1.4.1 MIN-MAX . . . . .	15
1.4.2 Trilaterazione . . . . .	16
1.4.3 Triangolazione . . . . .	17
1.5 Sensor Data Fusion . . . . .	18
<b>2 Descrizione del lavoro</b>	<b>20</b>
2.1 Livello black box . . . . .	21
2.2 Livello sottosistemi . . . . .	22
2.3 Livello moduli . . . . .	25
<b>3 Unità di misura inerziale</b>	<b>29</b>
3.1 Accelerometro . . . . .	29
3.2 Giroscopio . . . . .	31
3.3 Magnetometro . . . . .	32
3.4 Modello di misura . . . . .	34

<b>4 Stima dell'assetto tramite sensor fusion</b>	<b>38</b>
4.1 Rappresentazione geometrica dell'assetto di un corpo rigido nello spazio . . . . .	38
4.1.1 Matrice di rotazione . . . . .	40
4.1.2 Angoli di Eulero . . . . .	40
4.1.3 Quaternioni unitari . . . . .	43
4.2 Algoritmo di sensor fusion per la stima dell'assetto . . . . .	44
4.2.1 Equazioni di stato del sistema . . . . .	45
4.2.2 Filtro di Kalman a due stadi . . . . .	46
4.2.3 Descrizione dell'algoritmo . . . . .	47
4.2.4 Trasformazione del sistema a tempo discreto . . . . .	48
4.2.5 Riepilogo algoritmo step-by-step . . . . .	49
4.2.6 Settaggio iniziale delle costanti . . . . .	50
<b>5 Implementazione</b>	<b>52</b>
5.1 Implementazione Hardware . . . . .	52
5.1.1 Schema hardware del sistema . . . . .	52
5.1.2 Canali di comunicazione . . . . .	54
5.1.2.1 I2C . . . . .	55
5.1.2.2 USB CDC . . . . .	57
5.2 Implementazione Software . . . . .	61
5.2.1 Diagramma Comportamentale . . . . .	61
5.2.1.1 Diagramma del Microcontrollore . . . . .	61
5.2.1.2 Diagramma del modulo APP . . . . .	63
5.2.2 Modalità di computazione . . . . .	64
5.2.2.1 Low Computation mode . . . . .	64
5.2.2.2 High Computation mode . . . . .	68
5.2.2.3 Testing Computation mode . . . . .	69
<b>6 Analisi e validazione dei risultati</b>	<b>74</b>
6.1 Analisi temporale . . . . .	74
6.1.1 Analisi del tempo di lettura dei dati grezzi dall'IMU . . . . .	75
6.1.2 Analisi del tempo di trasmissione dei dati tramite USB . . . . .	76
6.1.3 Conclusioni analisi temporale . . . . .	77
6.2 Analisi dello zero-offset . . . . .	79
6.2.1 Stima dello zero-g offset . . . . .	79

6.2.2	Stima dello zero-rate offset . . . . .	82
6.2.3	Conclusioni stima dello zero-offset . . . . .	84
6.3	Analisi qualitativa dell'assetto stimato . . . . .	84
<b>7</b>	<b>Conclusioni e prospettive future</b>	<b>88</b>
<b>A</b>	<b>Script</b>	<b>89</b>
A.1	Operativi . . . . .	89
A.1.1	Script per la lettura dei valori misurati da Accelerometro e Giroscopio tramite I2C . . . . .	89
A.1.2	Script di inizializzazione del micro . . . . .	91
A.1.3	Script per la stima del tempo di trasmissione di dati al modulo App . . . . .	92
A.1.4	Script Low Computation Mode . . . . .	93
A.1.5	Script High Computation Mode . . . . .	95
A.1.6	Script Test Computation Mode per il modulo Microcontrollore . . . . .	97
A.1.7	Script Test Computation Mode per il modulo APP . . . . .	100
A.2	Testing . . . . .	103
A.2.1	Script per la stima del tempo di lettura di dati provenienti dall'IMU . . . . .	103
A.2.2	Script per la stima del tempo di trasmissione di dati al modulo App . . . . .	104
A.2.3	Script per il sistema ausiliario Arduino per l'analisi qualitativa	105
<b>B</b>	<b>Filtro di Kalman</b>	<b>107</b>
B.1	Descrizione teorica del filtro di Kalman . . . . .	107
B.2	L'algoritmo del filtro di Kalman discreto . . . . .	109
B.3	Il filtro di Kalman esteso . . . . .	112
<b>Bibliografia</b>		<b>116</b>

# Elenco delle figure

1	Planimetria originale . . . . .	1
2	Planimetria alterata . . . . .	1
3	Rete allo step 1 dell'esploratore . . . . .	2
4	Rete allo step 2 dell'esploratore . . . . .	2
5	Rete allo step 3 dell'esploratore . . . . .	2
6	Rete allo step 4 dell'esploratore . . . . .	2
7	Rete allo step 5 dell'esploratore . . . . .	3
8	Rete allo step 6 dell'esploratore . . . . .	3
1.1	Esempio esplicativo del concetto di contesto . . . . .	5
1.2	Sondaggio tra 74 casi di studio di applicazioni IPS [8] . . . . .	7
1.3	RSSI - Andamento della potenza in funzione della distanza percorsa dal segnale . . . . .	9
1.4	ToA - Princípio di funzionamento . . . . .	11
1.5	TDoA - Princípio di funzionamento . . . . .	12
1.6	TDoA - Stima della posizione lungo l'iperbole identificata da due nodi . . . . .	13
1.7	AoA - Stima della posizione attraverso l'angolo di incidenza . . . . .	14
1.8	MIN-MAX - Tecnica di posizionamento . . . . .	16
1.9	Trilaterazione - Esempio esplicativo . . . . .	16
1.10	Triangolazione - Esempio esplicativo . . . . .	18
2.1	Rappresentazione dei livelli d'astrazione utilizzati per descrivere il sistema . . . . .	20
2.2	Rappresentazione del sistema come black box . . . . .	21
2.3	Ambiguità nella georeferenziazione di un secondo nodo utilizzando soltanto la distanza dal precedente . . . . .	23

2.4	Rappresentazione dei sottosistemi necessari per georeferenziare i nodi della rete . . . . .	24
2.5	Rappresentazione dei sottosistemi in moduli . . . . .	25
2.6	Rappresentazione del flusso di dati tra i moduli dei sottosistemi, step 1 . . . . .	26
2.7	Rappresentazione del flusso di dati tra i moduli dei sottosistemi, step 2 . . . . .	27
2.8	Rappresentazione del flusso di dati tra i moduli dei sottosistemi, step 3 . . . . .	27
3.1	Rappresentazione esemplificativa di un accelerometro capacitivo a riposo . . . . .	30
3.2	Rappresentazione esemplificativa di un accelerometro capacitivo che subisce una forza esterna . . . . .	31
3.3	Rappresentazione di un giroscopio vibrante per la misura della velocità angolare lungo l'asse z . . . . .	32
3.4	Rappresentazione esemplificativa di un magnetometro capacitivo lungo l'asse z . . . . .	34
3.5	in 3.5(a) il <i>b-frame</i> nell'istante $t_1$ e $t_2$ relativamente al <i>n-frame</i> , in 3.5(b) l' <i>n-frame</i> in latitudine $\varphi$ e longitudine $\lambda$ , l' <i>e-frame</i> all'angolo $\alpha(t) = \omega_{iet}$ e l' <i>i-frame</i> . . . . .	35
4.1	Rappresentazione degli angoli di roll, pitch e yaw per un velivolo [22]	41
4.2	Significato geometrico degli angoli di Eulero [21] . . . . .	42
4.3	Rappresentazione del fenomeno di <i>gimbal lock</i> [23] . . . . .	43
4.4	Rappresentazione del principio di funzionamento del filtro di Kalman a due stadi [24] . . . . .	46
5.1	Schema hardware del sistema . . . . .	53
5.2	Foto del canale di comunicazione I2C tra il microcontrollore e l'unità di misura inerziale. . . . .	55
5.3	Rappresentazione di dispositivi collegati mediante I2C . . . . .	55
5.4	Rappresentazione del flusso di dati generato per la lettura, tramite I2C, di un registro ad 8 bit. . . . .	56
5.5	Foto del canale di comunicazione USB usato nella classe CDC per la trasmissione dei dati dal microcontrollore all'elaboratore. . . . .	58

5.6	Rappresentazione di un dispositivo CDC che supporta tre diverse funzionalità . . . . .	59
5.7	Rappresentazione del comportamento per il modulo Microcontrollore .	62
5.8	Rappresentazione del comportamento per il modulo Microcontrollore .	63
5.9	Rappresentazione del pacchetto dati utilizzato in <i>Low Computation Mode</i> a 6DOF . . . . .	64
5.10	Rappresentazione del pacchetto dati utilizzato in <i>Low Computation Mode</i> a 9DOF . . . . .	66
5.11	Rappresentazione del pacchetto dati utilizzato in <i>High Computation Mode</i> . . . . .	68
5.12	Rappresentazione del pacchetto dati utilizzato in <i>Test Computation Mode</i> a 6DOF . . . . .	69
5.13	Rappresentazione del pacchetto dati utilizzato in <i>Test Computation Mode</i> a 9DOF . . . . .	71
6.1	In 6.1(a) le operazioni basilari per la modalità <b>LCM</b> , in 6.1(b) le operazioni basilari per le modalità <b>HCM</b> e <b>TCM</b> . . . . .	75
6.2	in 6.2(a) l'IMU posto parallelamente all'asse X, in 6.2(b) l'IMU posto parallelamente all'asse Y e infine in 6.2(c) l'IMU posto parallelamente all'asse Z . . . . .	80
6.3	valori degli zero-g offset calcolati rispettivamente sull'asse X 6.3(a), sull'asse Y 6.3(b) e sull'asse 6.3(c) . . . . .	81
6.4	valori degli zero-rate offset per i campioni catturati in 2 minuti rispettivamente sull'asse X, Y e Z. . . . .	83
6.5	Schema hardware del sistema ausiliario realizzato per eseguire l'analisi qualitativa . . . . .	85
6.6	Risultati analisi qualitativa . . . . .	86
6.7	Retta di divergenza della stima dello yaw eseguita dal modulo APP. . . . .	87
B.1	Rappresentazione dell'algoritmo di Kalman per un sistema tempo discreto . . . . .	111
B.2	Rappresentazione dell'algoritmo di Kalman esteso per un sistema tempo discreto non lineare . . . . .	115

# Elenco delle tabelle

4.1	Tabella comparativa delle rappresentazioni d'assetto . . . . .	39
5.1	Tabella con gli usi tipici degli <i>endpoints</i> per comunicazioni USB-CDC	60
5.2	Tabella riepilogativa dei campi presenti nel pacchetto dati in LCM a 6DOF . . . . .	65
5.3	Tabella riepilogativa dei campi presenti nel pacchetto dati in LCM a 9DOF . . . . .	67
5.4	Tabella riepilogativa dei campi presenti nel pacchetto dati in HCM .	68
5.5	Tabella riepilogativa dei campi presenti nel pacchetto dati in TCM a 6DOF . . . . .	70
5.6	Tabella riepilogativa dei campi presenti nel pacchetto dati in TCM a 9DOF . . . . .	72
6.1	Media, Max e percentuali di occorrenze dei valori compresi nell'intervallo (0,1) e maggiori di 1 dei dati mostrati in Fig.6.3(a) . . . .	82
6.2	Media e varianza per i valori di zero-rate offset mostrati in Fig.6.4 .	83

# Introduzione

Il lavoro di questa tesi si colloca in un progetto finalizzato alla realizzazione di un sistema di geolocalizzazione di operatori in contesti privi di segnale GPS. Quest'ultimo permetterà ad un “esploratore” di creare dinamicamente una rete di nodi all'interno di zone nelle quali il segnale GPS è assente o comunque debole. Questa rete verrà poi ampliata ed utilizzata dagli operatori successivi ad esso per geolocalizzarsi all'interno della zona ed intervenire in maniera ottimale.

Uno scenario esemplificativo è quello di un vigile del fuoco che interviene in un edificio per soccorrere una persona. Tale operatore è considerato un esploratore, poiché è il primo ad intervenire e la planimetria dell'edificio risulta essere ignota e/o cambiata a seguito dell'evento disastroso (Fig.1 e Fig.2).

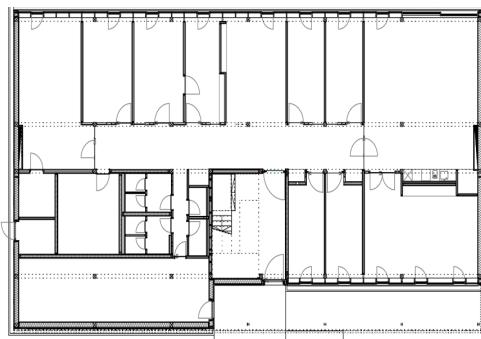


Figura 1: Planimetria originale

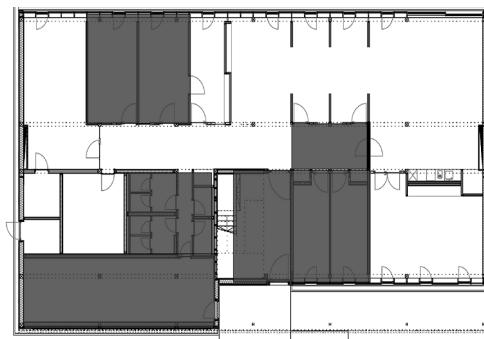


Figura 2: Planimetria alterata

I rettangoli in grigio (Fig.2) rappresentano aree non più accessibili dell’edificio, mentre le mura interrotte nuovi percorsi creati a causa dei crolli. Durante tutta la fase di *scouting* (esplorazione, ricerca, indagine), l’esploratore posizionerà un’ancora (nodo) ogni 20 metri approssimativamente e ogni qualvolta la precedente risulti non essere più in line-of-sight (linea visiva). Così facendo si “lascerà dietro una scia di briciole” che gli permetteranno di orientarsi all’interno dell’edificio, di eseguire il percorso all’ inverso o di ricevere supporto da un’ulteriore operatore. Osservando dalla Fig.3 alla Fig.8, si può notare come la rete cresca man mano che l’esploratore avanza all’interno dell’edificio.

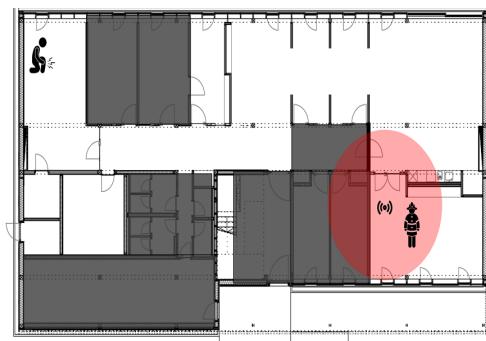


Figura 3: Rete allo step 1 dell’esploratore

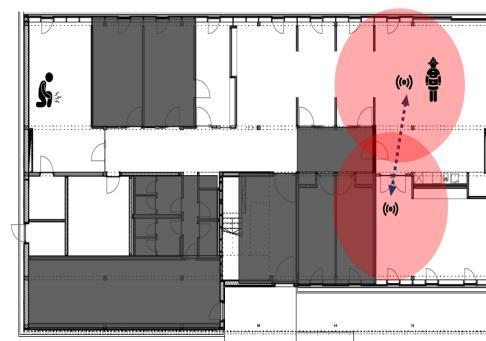


Figura 4: Rete allo step 2 dell’esploratore

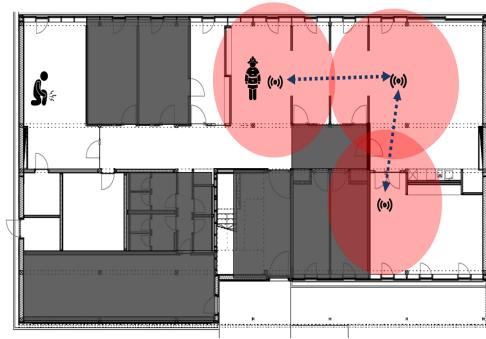


Figura 5: Rete allo step 3 dell’esploratore

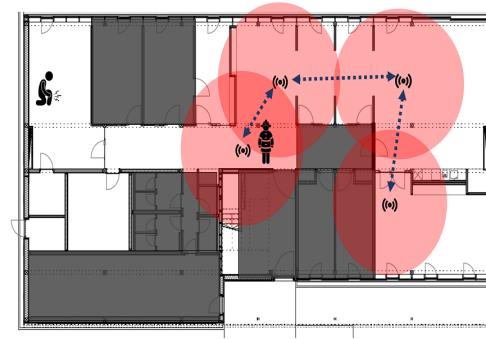


Figura 6: Rete allo step 4 dell’esploratore

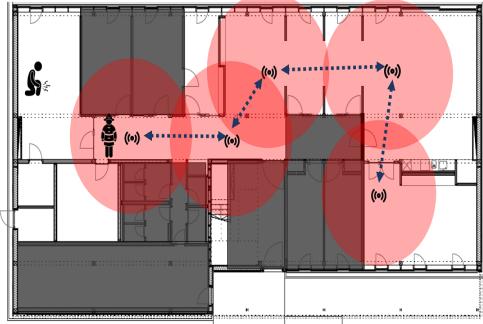


Figura 7: Rete allo step 5 dell'esploratore

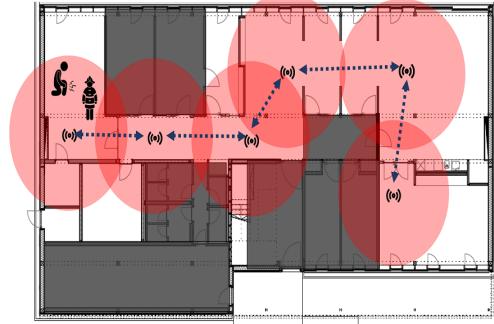


Figura 8: Rete allo step 6 dell'esploratore

Una volta creata l'infrastruttura, gli operatori potranno comunicare e condividere informazioni come posizione e stato.

La presente tesi è così strutturata:

**Capitolo 1:** Vengono descritti i problemi della geolocalizzazione indoor e lo stato dell'arte.

**Capitolo 2:** Viene descritto il lavoro seguendo un approccio top down.

**Capitolo 3:** Vengono illustrate le tecnologie adottate (IMU e sensori MEMS) e le problematiche legate ad esse.

**Capitolo 4:** Viene descritta l'elaborazione dei dati grezzi e l'algoritmo di sensor fusion implementato.

**Capitolo 5:** Si riportano alcuni dettagli riguardanti l'implementazione.

**Capitolo 6:** Vengono effettuate delle analisi e discussi i risultati ottenuti.

**Capitolo 7:** Conclusioni e prospettive future.

# Capitolo 1

## Contesto applicativo

Nella parte iniziale di questo capitolo viene esplicato il contesto applicativo di questa tesi e, una volta delineato vengono illustrate le tecniche basilari più utilizzate in questo ambito.

### 1.1 Context-aware computing

Il sistema realizzato nell'ambito di questa tesi, si colloca nel paradigma di computazione noto come *Context-aware computing*, ovvero un'applicazione nel quale i servizi utilizzano informazioni relative al contesto. In [1] si definisce come *context*:

*Ogni informazione che può essere usata per caratterizzare la situazione di un'entità. Ovvero una persona, un posto o un oggetto che è considerato rilevante all'interazione tra l'utente e l'applicazione, inclusi quest'ultimi.*

Questa definizione facilita il lavoro di progettazione e sviluppo di un'applicazione permettendo di identificare quali informazioni sono importanti e quali no. Si consideri un'applicazione nel quale l'utente deve registrare il peso degli oggetti presenti nel magazzino tramite una bilancia come mostrato dalla Fig.1.1:



Figura 1.1: Esempio esplicativo del concetto di contesto

Nello scenario descritto le *entità* sono rispettivamente utente e sistema, mentre due possibili informazioni riguardanti il contesto sono la presenza di altre persone e la posizione geografica del magazzino.

La presenza di altre persone nelle vicinanze non influisce sul compito dell’utente, quindi non può essere considerata come informazione contestuale. La posizione geografica del magazzino invece sì, infatti se quest’ultimo fosse situato in Italia il peso verrebbe calcolato in chilogrammi mentre se fosse situato negli USA verrebbe calcolato in libbre.

I sistemi che reperiscono, usano o interpretano queste informazioni contestuali sono detti *context-aware* e vengono definiti in [1] come:

*Un sistema è context-aware se usa il contesto per fornire informazioni rilevanti e/o servizi agli utenti, dove la rilevanza dipende dal compito degli utenti.*

Uno dei tipi di context-aware più utilizzati si basa sul contesto della localizzazione, ovvero servizi basati sulla conoscenza di dove qualcosa o qualcuno si trovi. Nell’era moderna i servizi basati sulla localizzazione (in inglese: *Location based services* LBSs) stanno assumendo sempre più importanza nelle attività quotidiane dell’uomo grazie alle molteplici possibili applicazioni, tra le quali navigazione assistita per autoveicoli, tracking di persone sensibili (bambini, anziani, malati), servizi di emergenza e così via.

I LBSs vengono divisi in due macro categorie:

- **OPSs:** *Outdoor Positioning System Services*, ovvero servizi di localizzazione in ambienti aperti.
- **IPSS:** *Indoor Positioning System Services*, ovvero servizi di localizzazione in ambienti indoor.

La tecnologia satellitare nota come Global Positioning System (GPS) è la tecnologia dominante negli OPSs. Attraverso una rete dedicata di satelliti artificiali in orbita, fornisce ad un terminale mobile o ricevitore GPS informazioni sulle sue coordinate geografiche ed orario, in ogni condizione meteorologica, ovunque sulla Terra o nelle sue immediate vicinanze ove vi sia un contatto privo di ostacoli con almeno quattro satelliti del sistema. La localizzazione avviene tramite la trasmissione di un segnale radio da parte di ciascun satellite e l'elaborazione dei segnali ricevuti da parte del ricevitore [2].

Il grande limite di questa tecnologia è che i ricevitori devono essere nella *line of sight* (letteralmente a vista d'occhio) di almeno quattro satelliti nel cielo, questo significa che all'interno di edifici e spazi chiusi il segnale viene attenuato e i sistemi perdono di accuratezza. Quindi la tecnologia GPS non è adatta ai servizi di localizzazione indoor.

Il sistema realizzato nell'ambito di questa tesi si colloca nell'ambito degli IPSS approfonditi nel paragrafo successivo.

## 1.2 Indoor Positioning System Service

Un sistema di posizionamento indoor (in inglese: Indoor positioning system o IPS) è un sistema in grado di localizzare *oggetti* o *persone* all'interno di edifici utilizzando onde radio, campi magnetici, segnali acustici e/o altre informazioni raccolte dai sensori all'interno di dispositivi mobili [3] o da altri appositamente installati nell'ambiente. Questi sono una specializzazione dei più generici sistemi *Real Time Locating Systems* (RTLS), standardizzati dall'*International Organization for Standardization and the International Electro Technical Commission* (ISO/IEC 24730).

Lo standard definisce i sistemi RTLS come:

*“ I Real Time Locating System sono sistemi wireless con l'abilità di localizzare la posizione di oggetti ovunque essi siano (in uno spazio definito) con un tempo di risposta che è, o si avvicina, a quello real time. La posizione è derivata dalla misurazione delle proprietà fisiche del collegamento radio.”*

La differenza tra RTLS e IPS è che gli IPS sono pensati per essere utilizzati da utenti su dispositivi mobili per navigare, orientarsi ed essere tracciati all'interno di edifici, mentre gli RTLS sono più generici e includono anche sistemi di localizzazione real-time che forniscono LBSs di tipo OPSs (Cap.1.1).

Come già accennato, gli IPS [4] permettono di creare una vasta gamma di servizi, ad esempio:

- **Way-Finding:** permettere di navigare in edifici complessi, come ad esempio aeroporti, seguendo il percorso indicato.
- **Ricerca dei punti d'interesse,** aumentare la customer experience facendo trovare all'utente ciò che desidera.
- **Multi-Dot:** visualizzare in una mappa le posizioni degli utenti per tracciare persone potenzialmente in pericolo (bambini, anziani).
- **Marketing di prossimità:** realizzare marketing mirati, inviando annunci sulle ultime offerte.

L'elenco di cui sopra rappresenta solo un ridotto sottoinsieme dei potenziali campi applicativi (vedi Fig.1.2), per questo motivo negli ultimi anni [7] l'interesse nella ricerca e nello sviluppo di sistemi di questo tipo è cresciuto sempre più tra le aziende, che hanno percepito la possibilità di grandi profitti in un mercato non ancora del tutto esplorato.

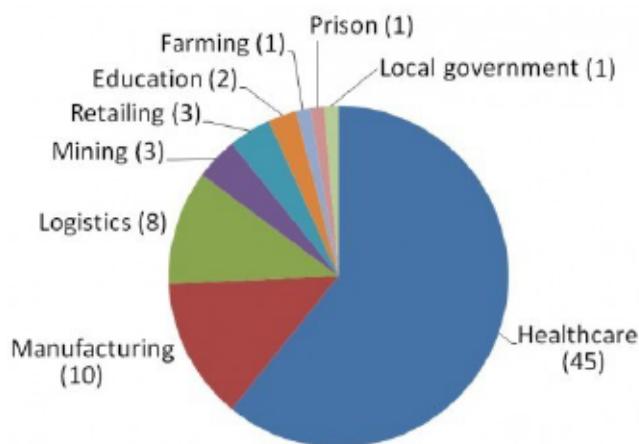


Figura 1.2: Sondaggio tra 74 casi di studio di applicazioni IPS [8]

Secondo un sondaggio di *Markets and Markets* e un articolo pubblicato da *The International News Magazine*, il mercato degli IPS subirà una crescita annuale media del 42.1% arrivando a valere 2.60 bilioni di dollari nel 2018. Questo da un'idea del perché grandi aziende come Google, Sony, Microsoft e Apple stiano investendo in questo settore.

Un'ulteriore spinta è data dal fatto che, al contrario del GPS per gli OPS (Cap.1.1), tuttora non esiste uno standard di riferimento per gli IPS. Infatti sul mercato sono disponibili diversi tipi di IPS commerciali che si differenziano in base al principio di funzionamento e alle tecniche utilizzate, utilizzando hardware specifico o la combinazione di più sistemi.

### 1.3 Stima della posizione

Gli IPS possono essere classificati sulla base di diversi fattori, uno di questi è su come determinano la posizione dei nodi.

Stimare [4] la distanza tra dispositivi wireless è utile perché attraverso questa informazione è possibile determinarne (con un certo errore) la posizione di un ricevitore rispetto ad un trasmettitore ([5], [6]), queste tecniche si distinguono in:

- **Range based:**
  - *RSSI*- potenza del segnale radio ricevuto(sez.1.3.1.1)
  - *ToA* - tempo d'arrivo: (sez.1.3.1.2)
  - *TDoA* - differenze del tempo di arrivo (sez.1.3.1.3);
- **Angle based:**
  - *AoA* - Angle of Arrival (sez.1.3.2.1).

Per poter determinare le distanze si devono distinguere i punti di riferimento (che hanno delle coordinate note) dai nodi senza posizione nota a cui assegnare delle coordinate. Si dicono:

- **Anchor**: i nodi le cui coordinate sono note
- **Target**: il nodo di cui non si conosce la posizione.

L’obiettivo del posizionamento è assegnare le giuste coordinate agli *Unknown* rispetto ad un sistema di riferimento. Questo è strettamente legato all’implementazione dell’IPS e così anche la codifica delle posizioni all’interno del sistema di riferimento, infatti le coordinate potrebbero essere restituite all’utente in maniera relativa (“vicino alla cucina”) oppure assoluta (“tre metri in direzione ovest dal nodo 1”).

### 1.3.1 Range based

Nel posizionamento dei nodi basato sulla distanza la stima della posizione del target dipende dai seguenti parametri:

- il tempo trascorso tra l’emissione e la ricezione del segnale radio;
- la distanza euclidea tra ogni emettitore ed il ricevitore;
- la potenza del segnale ricevuto.

In alcuni casi sono necessarie tre o più *Anchor* per ottenere le coordinate da assegnare allo *Unknown*.

#### 1.3.1.1 Received Signal Strength Indicator - RSSI

La comunicazione [4] tra dispositivi wireless (senza fili) avviene tramite lo scambio di segnali propagati nell’aria. Durante la propagazione i segnali tendono ad attenuarsi con l’aumentare della distanza percorsa fino a non essere più percepibili.

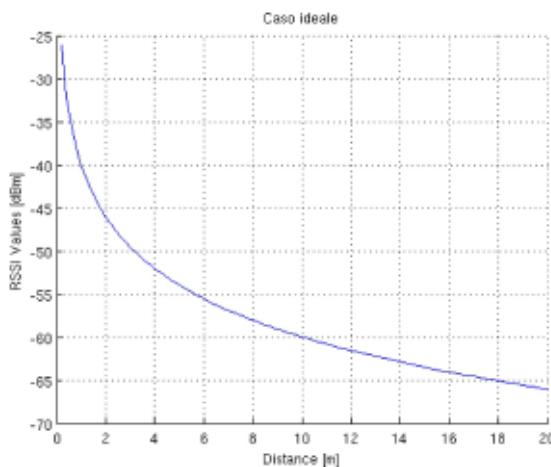


Figura 1.3: RSSI - Andamento della potenza in funzione della distanza percorsa dal segnale

La stima della potenza del segnale ricevuto è data dall'indicatore RSSI [9]. La distanza emettitore-ricevitore si stima utilizzando **l'equazione di trasmissione di Friis**.

$$P_T = P_R \frac{G_T G_R \lambda^2}{(4\pi)^2 d^n} \quad (1.1)$$

dove:

- $P_R$ : potenza del segnale ricevuto (Watt)
- $P_T$ : potenza del segnale trasmesso(Watt)
- $G_R$ : guadagno dell'antenna ricevente
- $G_T$ : guadagno dell'antenna trasmittente
- $\lambda = \frac{c}{f}$ : lunghezza d'onda, dove  $c$  è la velocità di propagazione e  $f$  è la frequenza dell'onda
- $d$ : distanza espressa in metri
- $n$ : costante di propagazione del segnale che dipende dall'ambiente

Con la seguente equazione invece è possibile convertire la potenza espressa in Watt nella potenza espressa in dBm:

$$P[dBm] = 10 \log_{10}(10^3 P[W]) \quad (1.2)$$

Combinando l'equazione 1.1 con 1.2 e applicando le proprietà dei logaritmi si ottiene:

$$RSSI = -(10n \log_{10} d - A) \quad (1.3)$$

dove  $A$  è la potenza del segnale ricevuto a distanza fissa di un metro (espressa in dBm), considerando una costante di propagazione  $n$ .

La stima della distanza si ottiene infine dalla seguente equazione:

$$d = 10^{(\frac{A - RSSI}{10n})} \quad (1.4)$$

Tuttavia la distanza restituita non è del tutto precisa, infatti la potenza del segnale potrebbe essere alterata dall'ambiente circostante attraverso i fenomeni di **Riflessione** (il segnale si riflette su vari ostacoli seguendo più percorsi) e di **Assorbimento** (il decadimento viene alterato dagli oggetti presenti). Tale tecnica viene solitamente completata utilizzando il metodo della **Trilaterazione** (sez.1.4.2)

### 1.3.1.2 Time Of Arrival measurements

A differenza del precedente metodo, con questa tecnica la distanza tra emettitore e ricevitore viene stimata sulla base del tempo impiegato dal segnale a raggiungere il ricevitore. Nello specifico la sequenza di azioni è:

1. Il nodo *A* invia il segnale al tempo  $t_1$
2. Il segnale arriva al nodo *B* al tempo  $t_2$
3. *B* elabora il messaggio impiegando un tempo  $t_d$  e lo invia al tempo  $t_3$
4. Il segnale torna al nodo *A* al tempo  $t_4$

Come mostrato dalla figura seguente:

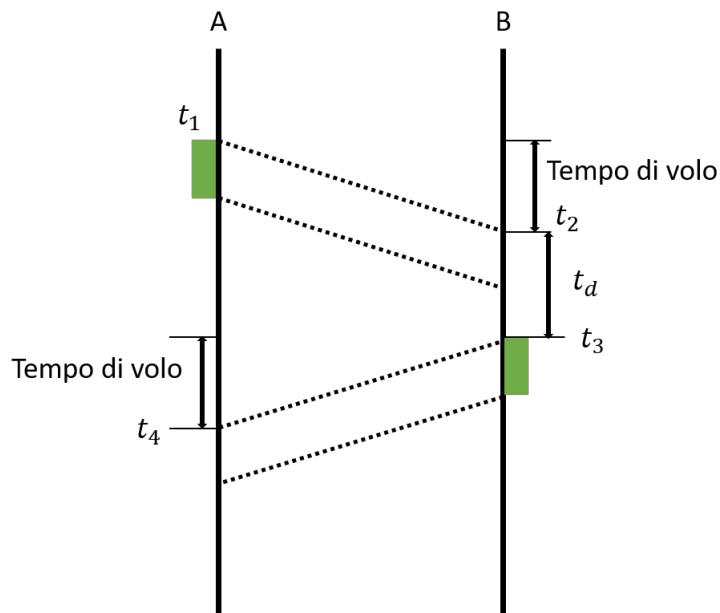


Figura 1.4: ToA - Principio di funzionamento

Quindi il tempo di volo può essere ricavato con la seguente equazione:

$$t_d = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \quad (1.5)$$

E infine la distanza stimata attraverso:

$$d_{ToA} = t_d * c \quad (1.6)$$

dove  $c$  è la velocità di propagazione della luce nel vuoto pari a 299792458 m/s. Per identificare in modo univoco un target, questa tecnica viene completata dalla tecnica di posizionamento nota come **Trilaterazione** (sez. 1.4.2), come per le misure RSSI viste precedentemente.

Il difetto principale di questa tecnica consiste nel fatto che sistemi utilizzati devono avere un complesso meccanismo di sincronizzazione per mantenere una fonte affidabile di tempo per i sensori[10].

### 1.3.1.3 Time Difference Of Arrival

Questa tecnica è basata sulla differenza nel tempo di arrivo di un segnale emesso da due sorgenti diverse verso un altro nodo, come mostrato in Fig.1.5).

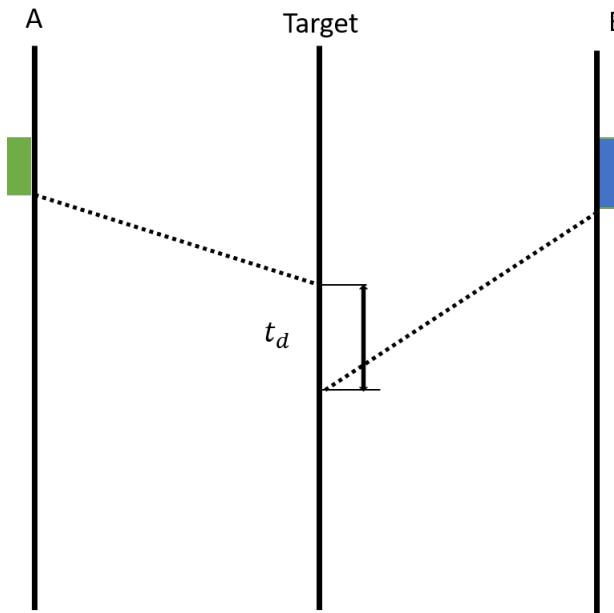


Figura 1.5: TDoA - Principio di funzionamento

Se si suppongono note le posizioni dei nodi  $A$  e  $B$  rispetto ad un sistema di riferimento, indicate rispettivamente dalle tuple  $(x_B, y_B)$  e  $(x_A, y_A)$ , la distanza del nodo *Target* può essere stimata dalla seguente equazione:

$$\Delta d = \Delta t_d * c \quad (1.7)$$

Dove:

- $c$  è la velocità di propagazione della luce nel vuoto
- $\Delta t$  è la differenza del tempo di arrivo dei segnali emessi dai nodi  $A$  e  $B$
- $\Delta d$  è la distanza in due dimensioni:  $(\sqrt{(x_B - x)^2 + (y_B - y)^2} - \sqrt{(x_A - x)^2 + (y_A - y)^2})$

In questo modo, la posizione del *target* viene stimata all'interno del luogo geometrico dei punti del piano aventi come costante la differenza delle distanze tra i nodi, ovvero dall'iperbole avente come fuochi i nodi  $A$  e  $B$ . Come mostrato in Fig.

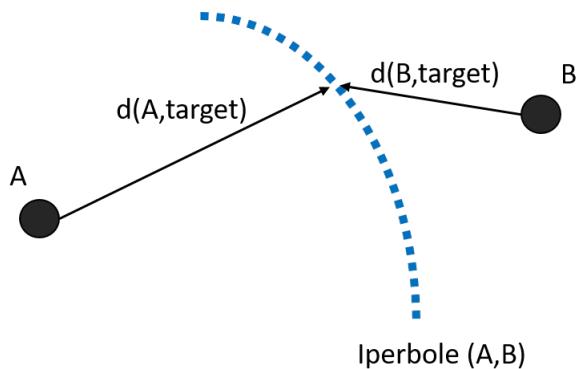


Figura 1.6: TDoA - Stima della posizione lungo l'iperbole identificata da due nodi

Tuttavia così facendo la posizione del *target* rimane stimata in un'insieme di punti infinito, quindi come per le tecniche viste precedentemente (sez.1.3.1.1 e sez.1.3.1.2) per identificare in modo univoco il target questa tecnica ha bisogno di essere completata dalla tecnica di posizionamento nota come **Trilaterazione** (sez.1.4.2).

### 1.3.2 Angle Based

#### 1.3.2.1 Angle of Arrival

Con questa tecnica la posizione del *target* viene stimata misurando gli angoli di incidenza del segnale trasmesso ad altri nodi.

Si consideri [11] un'antenna in un canale di propagazione, la tensione del segnale ricevuto (1.7) è data dalla seguente equazione:

$$V = \int_0^{2\pi} AoA(\varphi)G(\varphi)d\varphi \quad (1.8)$$

Dove:

- $AoA(\varphi)$  rappresenta l'ampiezza e la fase dell'onda incidente
- $G(\varphi)$  è il campo elettrico del nodo target
- $C$  valore proporzionale costante

Se si ruota l'antenna di un angolo  $\alpha$  intorno a se stessa nel piano cartesiano la precedente equazione diventa:

$$V(\alpha) = \int_0^{2\pi} AoA(\alpha - \varphi)G(\varphi)d\varphi \quad (1.9)$$

Possiamo notare che l'Eq.1.9 è la convoluzione di  $AoA$  e  $G$  e può essere scritta nel seguente modo:

$$V(\alpha) = CAoA(\alpha) * G(\alpha) \quad (1.10)$$

Si è scelto di normalizzare l'Eq.1.10 con il valore costante  $C$ . Quindi, utilizzando la transformata di Fourier l'Eq.1.10 diventa:

$$F(V(\alpha)) = F(AoA(\alpha))F(G(\alpha)) \quad (1.11)$$

Da cui è possibile calcolare l'angolo di incidenza desiderato:

$$AoA(\alpha) = F^{-1}\frac{F(V(\alpha))}{F(G(\alpha))} \quad se \quad F(G(\alpha)) \neq 0 \quad (1.12)$$

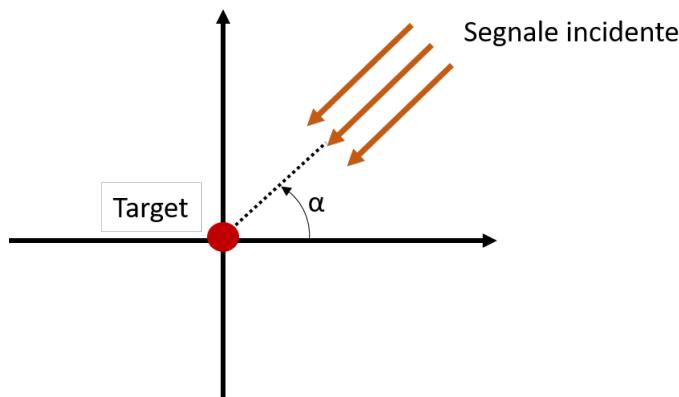


Figura 1.7: AoA - Stima della posizione attraverso l'angolo di incidenza

Il vantaggio dell'AoA risiede nella possibilità di ottenere un risultato attendibile senza la necessità di informazioni riguardanti i tempi di trasmissione. A fronte del

risparmio dal punto di vista computazionale, la tecnica presenta alcuni svantaggi pratici, dovuti al costo dell'hardware, che al fine di restituire informazioni precise, deve essere di alta qualità; rischiando altrimenti di incorrere in fenomeni che comprometterebbero la misurazione. Per questo motivo solitamente questa tecnica viene completata dalla tecnica di posizionamento nota come *Triangolazione* (sez.1.4.3).

## 1.4 Tecniche di localizzazione

Per tecniche di localizzazione si intendono tutte quelle tecniche che, combinate alle differenti metodologie di stima della posizione viste precedentemente (vedi 1.3), permettono di localizzare un nodo *target* all'interno di un sistema di riferimento. In questo paragrafo vengono illustrate quelle più conosciute e basilari nell'ambito degli IPS.

### 1.4.1 MIN-MAX

Combina le stime della distanza di più *anchor*, ottenute attraverso tecnica *RSSI* (sez.1.3.1.1), nel seguente modo:

- Stimare la distanza  $d_i$  di ogni nodo i-esimo in base al valore RSSI
- Traccia due linee orizzontali e verticali a distanza  $d_i$  dal nodo *target*
- Identifica un quadrato di lato  $2 d_i$  i cui estremi saranno:  
$$[max(x_i - d : i), max(y_i - d_i)] * [min(x_i + d_i), min(y_i + d_i))]$$
- Calcola le intersezioni dei quadrati

Il centro del quadrato (Fig.1.8) rappresenta la posizione stimata del *target*. Più piccola sarà l'area e maggiore sarà l'accuratezza della posizione stimata.

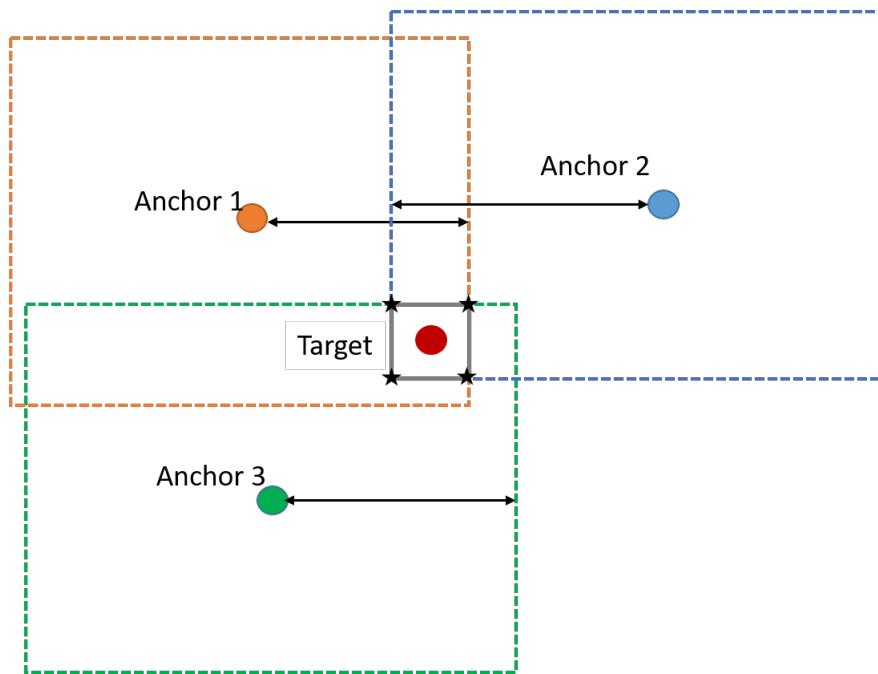


Figura 1.8: MIN-MAX - Tecnica di posizionamento

### 1.4.2 Trilaterazione

Consideriamo 3 *Anchor* intorno alle quali si disegnano 3 circonferenze aventi per centro le coordinate degli *Anchor* e per raggio l'RSSI del segnale ricevuto dallo *Unknown*, come mostrato in figura:

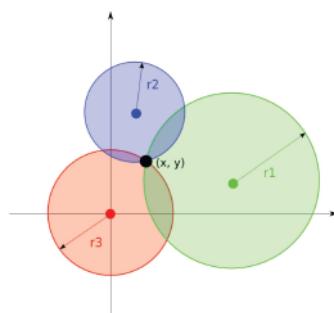


Figura 1.9: Trilaterazione - Esempio esplicativo

Quindi le coordinate dell' *Unknown* sono la soluzione del seguente sistema:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = r_2^2 \\ (x - x_3)^2 + (y - y_3)^2 = r_3^2 \end{cases} \quad (1.13)$$

In base alla soluzione del sistema si può avere una delle seguenti situazioni:

- La soluzione non è unica, si hanno tre cerchi che si sovrappongono
- Il sistema non ammette soluzione, i raggi vanno aumentati
- La soluzione esiste ed è unica, i tre cerchi si intersecano in un solo punto

### 1.4.3 Triangolazione

A differenza delle Trilaterazione (sez.1.4.2), queste tecniche identificano la posizione del nodo *target* a partire dagli angoli stimati da tre *anchors* attraverso una delle tecniche Angle Based (sez.1.3.2).

In [12] si descrive la triangolazione geometrica attraverso il seguente algoritmo:

1. Siano  $1, 2, 3$  le *anchor* in grado di stimare l'angolo, rispetto ad una circonferenza concentrica all'*anchor* stessa, del nodo target
2. siano  $L_{12}$  e  $L_{31}$  rispettivamente le distanze tra l'*anchor* 1 e 2 e l'*anchor* 3 e 1
3. Siano gli angoli compresi tra 1 e 2 e tra 1 e 3, indicati rispettivamente con  $\lambda_{12}$  e  $\lambda_{13}$ , minori di  $180^\circ$
4. sia  $\phi$  l'angolo tra l'asse x positivo e la linea formata dall'*anchor* 1 e 2
5. sia  $\sigma$  l'angolo tra l'asse x positivo, l'*anchor* 1 e l'*anchor* 3 più  $\phi$
6. sia  $\gamma = \sigma - \lambda_{31}$
7. sia  $p = \frac{L_{31} \sin \lambda_{12}}{L_{12} \sin \lambda_{31}}$
8. sia  $\tau = \tan^{-1} \frac{\sin \lambda_{12} - p \sin \gamma}{p \cos \gamma - \cos \lambda_{12}}$

$$9. \text{ sia } L_1 = \frac{L_{12} \sin(\tau + \lambda_{12})}{\sin \lambda_{12}}$$

Allora le coordinate  $x$  e  $y$  del target sono date da:

- $x_R = x_1 - L_1 \cos(\phi + \tau)$
- $y_R = y_1 - L_1 \sin(\phi + \tau)$
- $\Phi_R = \phi + \tau - \lambda_1$

Dove  $\Phi_R$  rappresenta l'orientamento del nodo target, come mostrato in figura:

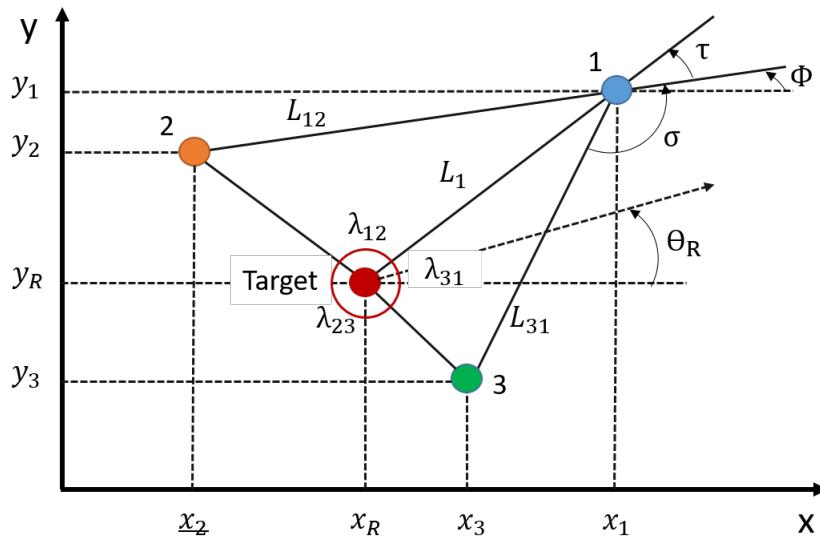


Figura 1.10: Triangolazione - Esempio esplicativo

## 1.5 Sensor Data Fusion

In generale con Sensor Data Fusion si indicano tutte quelle tecniche che [13] combinano dati provenienti da sensori o derivate da altre risorse tali che l'informazione risultante abbia una minore incertezza rispetto a quella ottenuta utilizzando le risorse individualmente.

Nel contesto degli IPS (1.2), il Sensor Fusion ha come scopo quello di determinare informazioni riguardanti la posizione di oggetti e/o persone. Le risorse delle informazioni grezze sono le più svariate, tra le quali:

- Accelerometro
- Giroscopio
- Magnetometro
- Infrarossi
- RFID
- Sensori ottici
- Sensori di pressione
- Bluetooth
- WiFi
- Telecamera
- UWB

Come e quali risorse utilizzare per stimare la posizione di un oggetto è una sfida ingegneristica non banale, ma grazie alla crescente richiesta di IPSs la ricerca di nuove tecnologie e algoritmi in questo settore ha permesso di raggiungere risultati incredibili.

# Capitolo 2

## Descrizione del lavoro

Per comprendere la soluzione proposta e il sistema ideato si utilizzerà un approccio top-down. Partendo da un livello d'astrazione tale in cui emergono solo i requisiti funzionali e il sistema è rappresentato da una *black box*, si raggiunge un livello che permetterà di affinare i requisiti iniziali e mostrare i *sottosistemi* che lo costituiscono. Infine nell'ultimo livello si avrà una visione dei *moduli* che compongono i sottosistemi e che sono stati realizzati nel contesto di questa tesi.

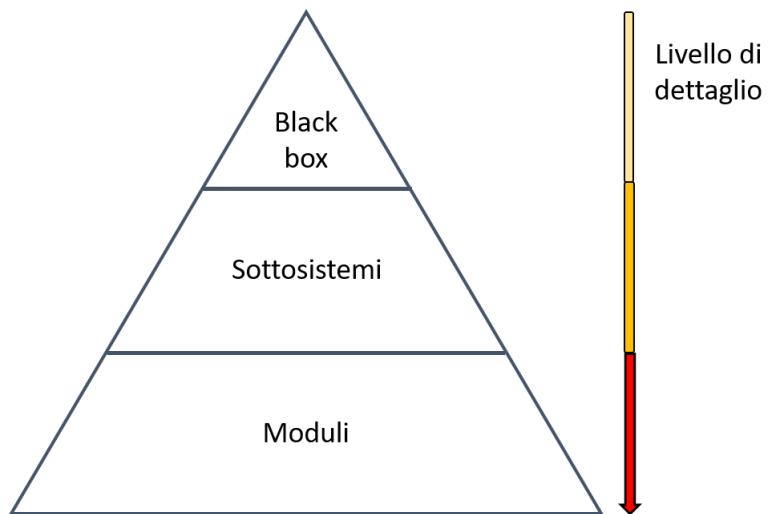


Figura 2.1: Rappresentazione dei livelli d'astrazione utilizzati per descrivere il sistema

## 2.1 Livello black box

Considerando il sistema in questione come una black box (Fig.2.2) e l'infrastruttura di rete in modo astratto, i due macro-requisiti funzionali sono rispettivamente:

- **R1:** Geolocalizzare l'operatore
- **R2:** Trasmettere messaggi predefiniti come stato della vittima e codici d'emergenza.

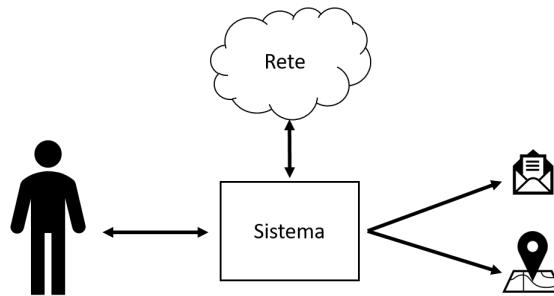


Figura 2.2: Rappresentazione del sistema come black box

Nelle fasi primordiali del progetto, si è scelto di allocare la maggior parte delle risorse lavorative nel completamento di **R1** lasciando ad una fase successiva lo sviluppo di **R2**. **R1** può essere suddiviso in due requisiti più specifici:

- **R1.1:** Determinare la posizione di un operatore all'interno della rete
- **R1.2:** Identificare il cammino minimo da un operatore ad un altro

Quest'ultimo rappresenta sia la possibilità da parte dell'operatore di eseguire il percorso all'inverso, sia la possibilità che venga raggiunto da una squadra di supporto. A questo punto si può scendere al livello d'astrazione successivo (2.1).

## 2.2 Livello sottosistemi

Per soddisfare il requisito **R1.1** è necessario che l'operatore si trovi all'interno di una zona *infrastrutturata*, ovvero una zona dove i nodi della rete siano georeferenziati rispetto ad un sistema di riferimento comune. Considerata la modalità con la quale la rete è costruita, ovvero dinamicamente da un'esploratore, garantire tale situazione non è un compito banale.

La soluzione si costruisce per iterazione georeferenziando i singoli nodi nel momento in cui vengono aggiunti dall'esploratore.

Con riferimento all'esempio illustrato precedentemente (Fig.3- Fig.8), si consideri lo step 2. Per ipotesi si supponga che il primo nodo sia già georeferenziato, nel momento in cui l'operatore risulti essere al limite della line-of-sight e/o della distanza di sicurezza piazzerà il secondo nodo.

Per mantenere il livello d'astrazione attuale basta sapere che le caratteristiche della tecnologia utilizzata nell'implementazione della rete, fa sì che le singole celle abbiano un raggio d'azione all'interno del quale il sottosistema *S1* (Fig.2.4) può calcolare la distanza tra l'operatore e il centro della cella di appartenenza.

Tale informazione non è però sufficiente, infatti ci sono infiniti punti sulla circonferenza con centro nel primo nodo e raggio pari alla distanza. Per poter georeferenziare in modo univoco il secondo nodo si deve trovare anche l'angolo in riferimento al primo nodo.

La figura 2.3 rappresenta un tentativo di georeferenziare il secondo nodo utilizzando soltanto la distanza tra i due nodi.

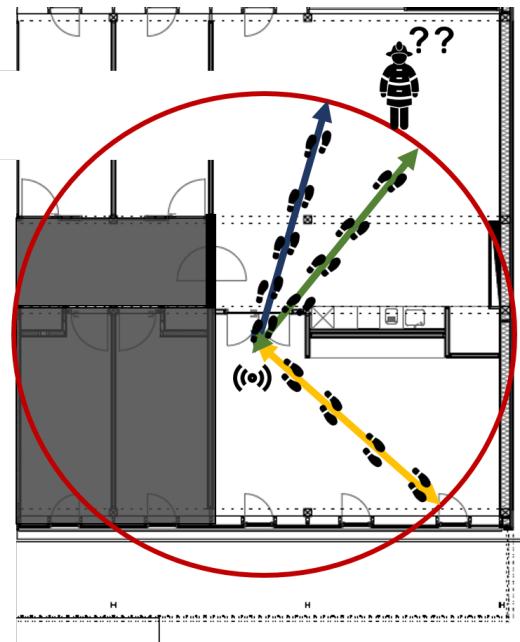


Figura 2.3: Ambiguità nella georeferenziazione di un secondo nodo utilizzando soltanto la distanza dal precedente

Nell'esempio appena proposto, si sono mostrate solo tre delle possibili infinite posizioni del secondo nodo. Si assuma che il punto corretto sia l'intersezione tra il vettore in blu e la circonferenza. In tal caso un'ambiguità con il vettore in verde potrebbe essere accettata in quanto si discosta di pochi metri dalla reale posizione, ben diversa sarebbe un'ambiguità con il vettore in giallo che renderebbe l'informazione del tutto errata e il sistema disinformato.

In conclusione per georeferenziare in maniera univoca il nuovo nodo (rispetto al primo) sono necessarie le seguenti informazioni:

- La distanza tra i due nodi
- L'angolo tra i due nodi

Per ricavare queste informazioni si sono ideati due sottosistemi, rappresentati dalla Fig.2.4:

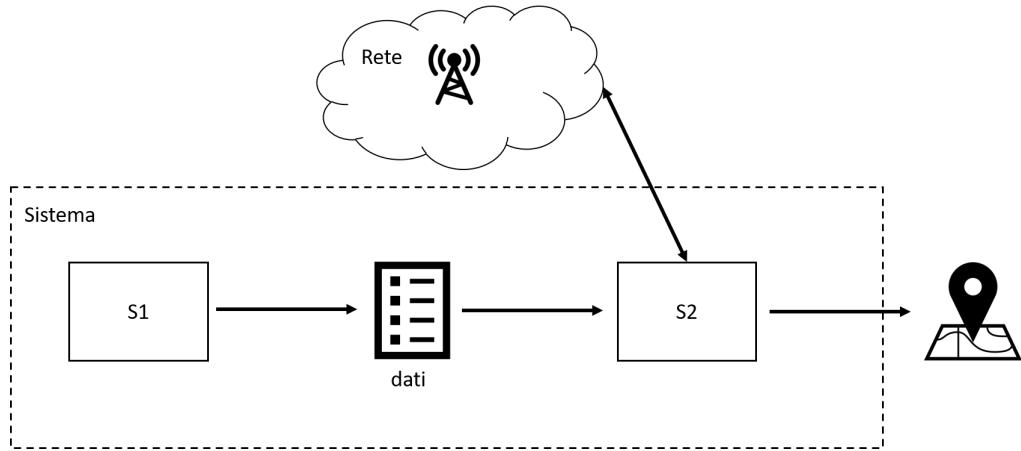


Figura 2.4: Rappresentazione dei sottosistemi necessari per georeferenziare i nodi della rete

Ognuno dei quali con la propria responsabilità:

- **S1:** ha il compito di ricavare tramite i sensori (cap.3) le informazioni riguardanti gli angoli assunti dall'esploratore lungo il tragitto
- **S2:** ha il compito di ricevere i dati da S1 e la distanza dal nodo precedente, elaborarli e infine determinare la posizione del nuovo nodo

Relativamente alla rappresentazione del sistema mostrata in Fig.2.4, il lavoro di questi tesi si colloca nella progettazione e nello sviluppo del sottosistema S1. Nel prossimo paragrafo si raggiungerà il livello d'astrazione più basso della piramide (vedi 2.1) dettagliando i moduli che compongono il sottosistema realizzato.

## 2.3 Livello moduli

Prima di "aprire" il sottosistema è bene specificare che si utilizzerà il termine *modulo* per riferirsi sia a componenti hardware che software. Questo abuso di notazione permetterà di mostrare con un unico livello d'astrazione tutti i moduli progettati e realizzati al fine di stimare gli angoli assunti dall'esploratore lungo il tragitto (si veda 2.2).

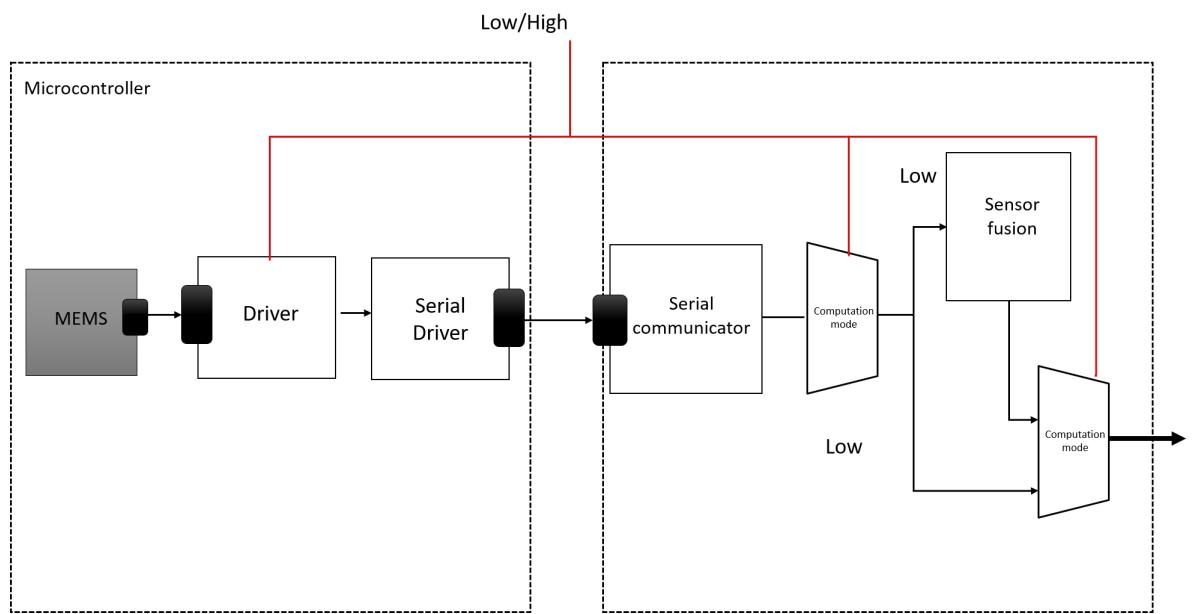


Figura 2.5: Rappresentazione dei sottosistemi in moduli

Per esplicare al meglio i compiti dei singoli moduli e mantenere l'attuale livello d'astrazione, fermo restando che tutti i dettagli tecnici verranno forniti nei capitoli successivi, si mostrerà il flusso di dati generato in uno qualsiasi degli intervalli di campionamento lungo il tragitto dell'operatore tra un nodo e l'altro.

Il flusso inizia nel momento in cui il modulo *driver* acquisisce le informazioni dall'unità *MEMS* (cap.3) riguardanti la velocità angolare, l'accelerazione e il campo magnetico relativi all'operatore. Come mostrato in Fig.2.6.

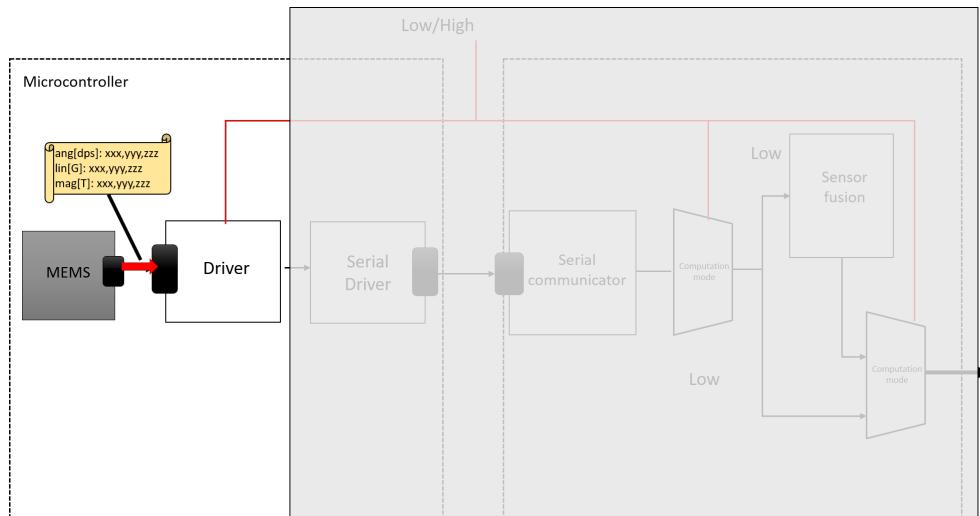


Figura 2.6: Rappresentazione del flusso di dati tra i moduli dei sottosistemi, step 1

I dati acquisiti sono "grezzi" e devono essere elaborati (Cap.4). Quando e da chi questa computazione verrà eseguita durante il flusso, viene stabilito attraverso il comando Low/High. Da questo ne consegue anche la modalità di funzionamento del driver in:

- **HCM:** High Computation mode, i dati vengono elaborati dal microcontrollore
- **LCM:** Low Computation mode, i dati verranno elaborati in seguito dal sottosistema *App*

La scelta tra quale di queste due modalità utilizzare verrà motivata nel capitolo riguardante l'analisi dei risultati (cap.4), per il momento si ipotizzi di settare la linea di comando al valore "Low" e quindi di utilizzare il driver in *LCM*. Con queste impostazioni i dati grezzi vengono impacchettati ed etichettati con un timestamp relativo, prima di essere inviati dal modulo *Serial driver* e ricevuti dal modulo *App* mediante il modulo *Serial communicator*, quest'ultimo li inoltra all'ingresso del multiplexer *Computation mode* come mostrato in Fig.2.7:

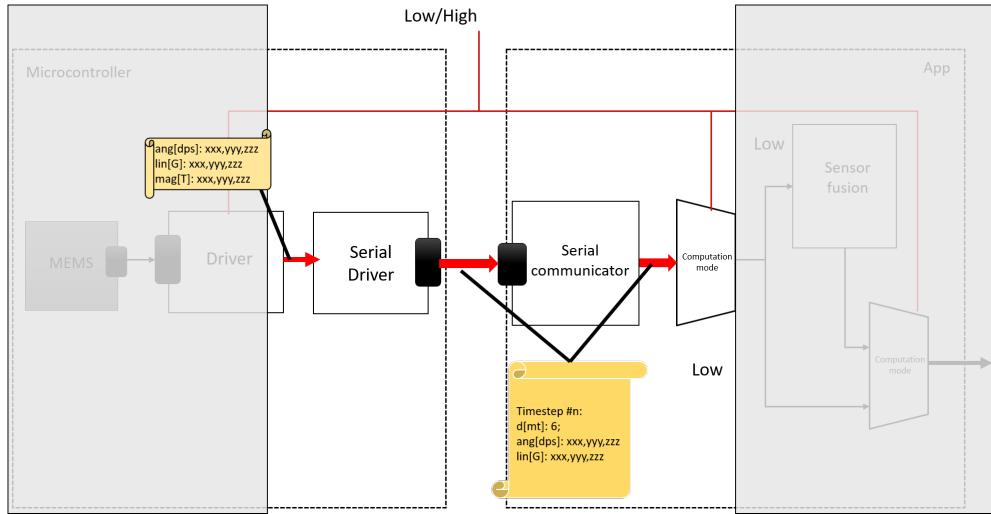


Figura 2.7: Rappresentazione del flusso di dati tra i moduli dei sottosistemi, step 2

Poiché per ipotesi si è scelto di settare la linea di comando sul valore **Low**, il multiplexer devia il flusso di dati verso il modulo *Sensor Fusion* come mostrato in figura:

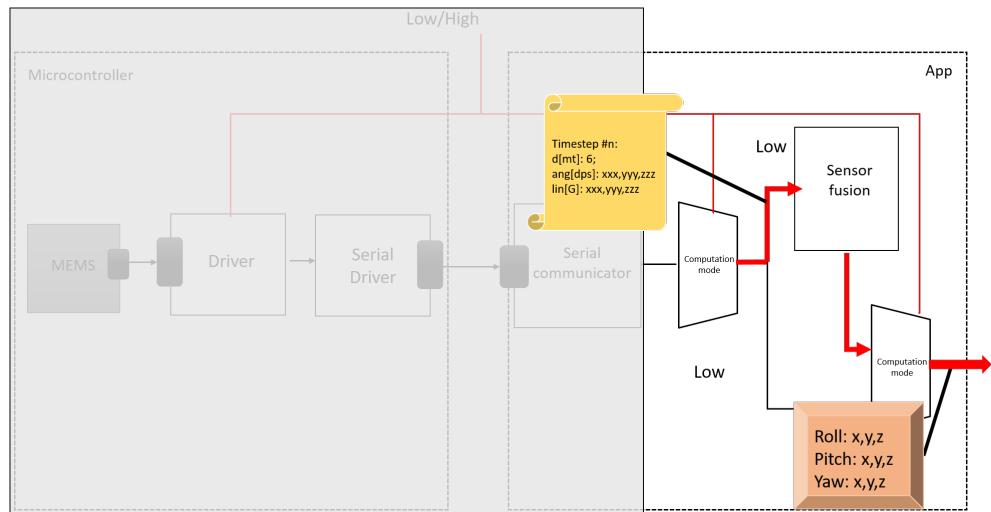


Figura 2.8: Rappresentazione del flusso di dati tra i moduli dei sottosistemi, step 3

A questo punto il modulo *Sensor Fusion* elabora i dati grezzi (vedi cap.4.2) fornendo in uscita gli angoli di Eulero (Cap.4.1) relativi all'operatore:

- **Roll**
- **Pitch**
- **Yaw**

Questi dati verranno usati come input dal sottosistema S2 (si veda 2.4) che provvederà, insieme all'informazione relativa alla distanza dal nodo precedente, a determinare l'angolo del nuovo nodo e quindi a risolvere il problema di georeferenziazione emerso nel livello di astrazione precedente (si veda 2.2).

# Capitolo 3

## Unità di misura inerziale

Le unità di misura inerziale [14] (in inglese *Inertial Measurement Units* - IMU) sono dispositivi elettronici basati su sensori inerziali come accelerometri (sez.3.1) e giroscopi (sez.3.2). In molti casi a questi vengono aggiunti altri sensori utili ad applicazioni di navigazione come il magnetometro (sez.3.3). Nello specifico di questa tesi, l'IMU utilizzata è un circuito integrato composto da questi tre sensori (più altri non utilizzati come sensore di temperatura) realizzati tramite tecnologia MEMS (acronimo di Microelectro Mechanical System, ovvero sistemi meccanici microelettrici).

Nel corso degli anni l'interesse per questa tecnologia è cresciuto grazie ai vantaggi in termini economici e tecnici, tra questi i più importanti sono:

- costo di realizzazione costante e proporzionale alla superficie del dispositivo
- grande potenziale di integrazione nei circuiti elettronici integrati
- basso consumo energetico
- dimensioni ridotte

In questo capitolo si illustrano i principi di funzionamento alla base dei sensori, realizzati mediante tecnologia MEMS, integrati nell'IMU utilizzata nel lavoro di questa tesi.

### 3.1 Accelerometro

In generale un accelerometro è un dispositivo in grado di misurare l'accelerazione di un corpo rigido causata da una forza esterna. Questa può essere statica, come la

forza di gravità, o dinamica nel caso di forze vibranti applicate al dispositivo. Uno dei più comuni accelerometri MEMS è quello *capacitivo* che, come il nome suggerisce, si basa sulla *capacità elettrostatica*. Se due piastre sono posizionate parallelamente tra di loro e poste ad una certa distanza, allora la capacità generata è data da:

$$C = \varepsilon_r \varepsilon_0 \frac{A}{d} \quad (3.1)$$

Dove:

- $C$  è la capacità
- $\varepsilon_0$  è la costante dielettrica del vuoto
- $\varepsilon_r$  è la costante dielettrica relativa al materiale utilizzato per le piastre
- $A$  è l'area delle piastre
- $d$  è la distanza tra le due piastre

Dall'equazione 3.1 si noti che la capacità può variare solo se vi sono cambiamenti nell'area delle piastre o nella loro distanza. Proprio su quest'ultimo parametro si basano gli accelerometri capacitivi. Una classica struttura è rappresentata dalla figura seguente:

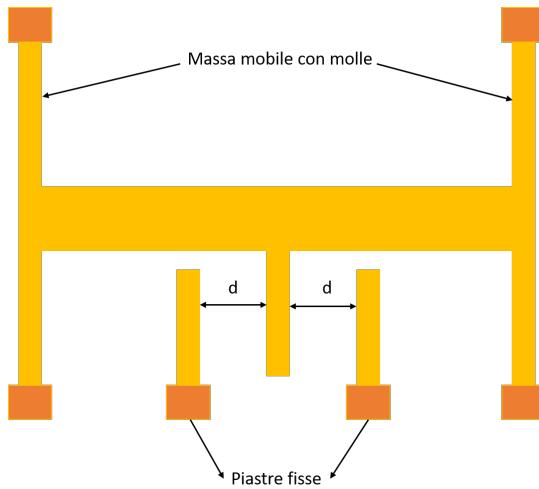


Figura 3.1: Rappresentazione esemplificativa di un accelerometro capacitivo a riposo

La massa centrale è in grado di muoversi lungo un asse orizzontale grazie a delle molle poste alle sue estremità, come rappresentato dalla figura seguente:

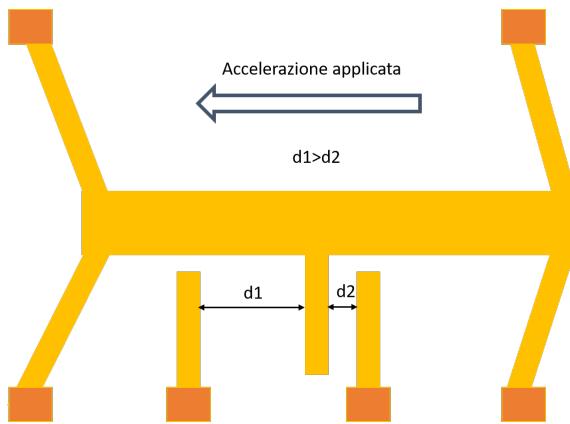


Figura 3.2: Rappresentazione esemplificativa di un accelerometro capacitivo che subisce una forza esterna

A seguito del movimento della massa centrale, la distanza  $d_2$  in Fig.3.2 si farà più piccola provocando una variazione di tensione ai capi della massa centrale. Questa verrà quindi convertita in un valore numerico rappresentante l'accelerazione subita in base alla scala e alla sensibilità del dispositivo.

## 3.2 Giroscopio

In generale, un giroscopio è un dispositivo in grado di misurare la velocità angolare a cui è sottoposto il dispositivo. I giroscopi realizzati mediante tecnologia MEMS si basano sulla forza di *Coriolis*. In fisica [18], la forza di Coriolis è una forza apparente a cui risulta soggetto un corpo quando si osserva il suo moto da un sistema di riferimento che sia in moto circolare rispetto ad un sistema di riferimento inerziale. I giroscopi di questo tipo sono composti [19] da una *massa*, due *molle* e due ammortizzatori come mostrato in Fig.3.3. Si assuma l'asse  $x$  come l'asse di direzione (*drive mode*) e l'asse  $y$  come l'asse di rilevamento(*sensing mode*). Quando la massa è sottoposta ad una vibrazione armonica applicata da una forza elettrostatica, elettromagnetica o elettrotermica lo spostamento lungo l'asse  $x$  è dato da:

$$x(t) = A_x \cos(\omega_x t) \quad (3.2)$$

Dove  $A_x$  è l'ampiezza e  $\omega_x$  è la frequenza angolare. Una velocità angolare  $\Omega_z$  in input intorno all'asse z causa un'accelerazione di Coriolis lungo l'asse y data dalla seguente equazione:

$$a_y = 2\Omega_z \times \frac{d_x}{dt} = -2\Omega_z A_x \omega_x \sin(\omega_x t) \quad (3.3)$$

La massa quindi inizierà a vibrare lungo l'asse y a causa della forza di Coriolis e la velocità angolare  $\Omega_z$  può essere calcolata misurando lo spostamento lungo l'asse vibrante.

Quando il *drive mode* e il *sense mode* sono perfettamente uguali ( $\omega_x = \omega_y$ ), l'ampiezza lungo l'asse y raggiunge il massimo mentre la larghezza di banda raggiunge il minimo. In generale, questi due parametri dovrebbero essere uguali al fine di ottimizzare la sensibilità e la larghezza di banda.

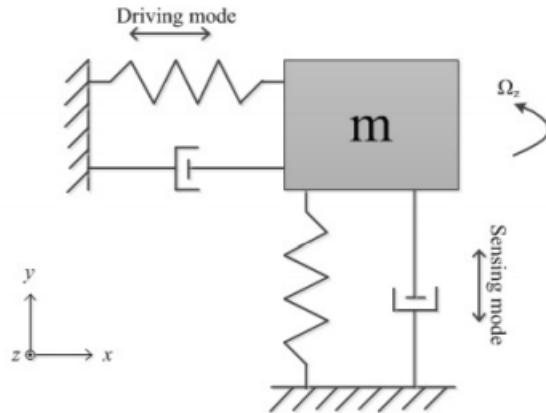


Figura 3.3: Rappresentazione di un giroscopio vibrante per la misura della velocità angolare lungo l'asse z

### 3.3 Magnetometro

In generale, un magnetometro è un dispositivo in grado di rilevare l'intensità e la direzione del campo magnetico presente. Questi si basano sulla ben nota forza di Lorentz.

Se si eroga una certa corrente in un conduttore posto in un campo magnetico trasversale al flusso, si genera una forza proporzionale alla velocità dei portatori, alla carica ed al valore del campo magnetico, diretta nella direzione ortogonale ad entrambi,

secondo la relazione:

$$\vec{F}_L = q \vec{v} \times \vec{B} \quad (3.4)$$

Dove:

- $q$  è la carica elementare
- $F$  è la forza di Lorentz
- $B$  è il campo magnetico nel vuoto

Indicando con  $l$  la lunghezza del conduttore si ha:

$$\vec{F}_L = l \vec{i} \times \vec{B} \quad (3.5)$$

Questa forza viene quindi sfruttata per misurare il campo magnetico esterno agente sul dispositivo. Una delle realizzazioni più comuni nell’ambito dei sensori realizzati mediante tecnologia MEMS sono i magnetometri capacitivi.

In Fig.3.4 è mostrata una semplice implementazione di un magnetometro capacitivo. Questo è composto da due *molle* i cui terminali sono ancorati al substrato del dispositivo dove viene fatta circolare una corrente elettrica. Nel punto centrale le molle mantengono sospeso un *rotore* dove idealmente non scorre corrente e al cui interno sono ancorati degli *statori*. In presenza di una forza di Lorentz, le *molle* si deformano dando origine ad uno spostamento rigido del *rotore*. Poiché lo *statore* non si muove, la capacità tra il *rotore* e lo *statore* cambia in maniera proporzionale all’intensità del campo magnetico esterno.

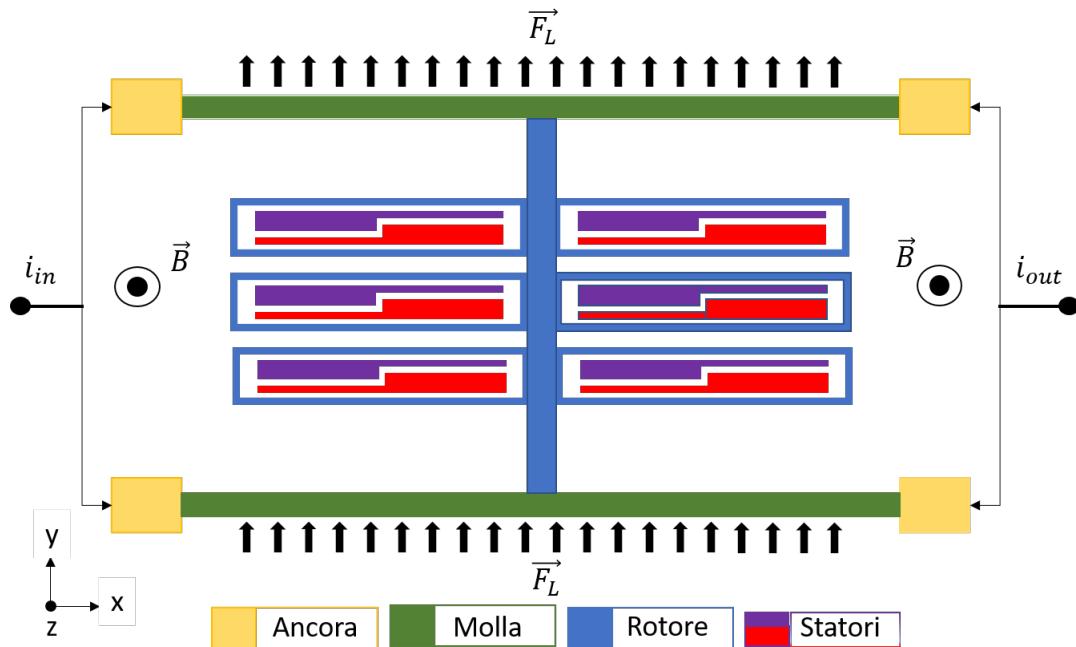


Figura 3.4: Rappresentazione esemplificativa di un magnetometro capacitivo lungo l'asse z

### 3.4 Modello di misura

Dopo aver introdotto i principi di funzionamento dei sensori è bene specificare il modello di misurazione.

I sensori utilizzati hanno tre assi lungo i quali una "quantità fisica" (esempio forza, velocità angolare, campo magnetico) è convertita in un segnale di tensione in uscita. Tipicamente questi sensori hanno un comportamento lineare nell'area di lavoro. Sulla base di questa osservazione, la seguente equazione (semplificata) descrive la relazione tra la forza fisica  $y(t)$  e la tensione in uscita dal sensore  $u(t)$ :

$$u(t) = GRy(t) + c \quad (3.6)$$

Dove:

- $G$  è la matrice diagonale contenente il guadagno per ogni asse sensibile
- $R$  è la matrice di allineamento che specifica la direzione degli assi
- $c$  è il vettore di offset

Al fine di discutere il modello di misura, si devono introdurre i seguenti sistemi di coordinate (in inglese: *coordinate frames*) rappresentati in Fig.3.4:

- Il **frame del corpo** - *b-frame* (in inglese: *body frame*): è il sistema di riferimento dei movimenti dell'IMU. L'origine è posta al centro dei sensori e allineata al case posto sul chip. Tutte le misure inerziali sono calcolate su questo sistema di riferimento.
- Il **frame di navigazione** - *n-frame* (in inglese: *navigation frame*) è il frame geografico nel quale vogliamo navigare. Per navigazioni a corto raggio è considerato statico rispetto alla terra.
- Il **frame inerziale** - *i-frame* (in inglese: *inertial frame*): è un frame stazionario non rotante. L'IMU misura le forze relativamente a questo frame. La sua origine è posta al centro della terra e i suoi assi sono allineati rispetto alle stelle.
- Il **frame terrestre** - *e-frame* (in inglese: *earth frame*): coincide con l' i-frame ma ruota intorno alla terra. L'origine è posta al centro della terra e gli assi fissati rispetto ad essa.

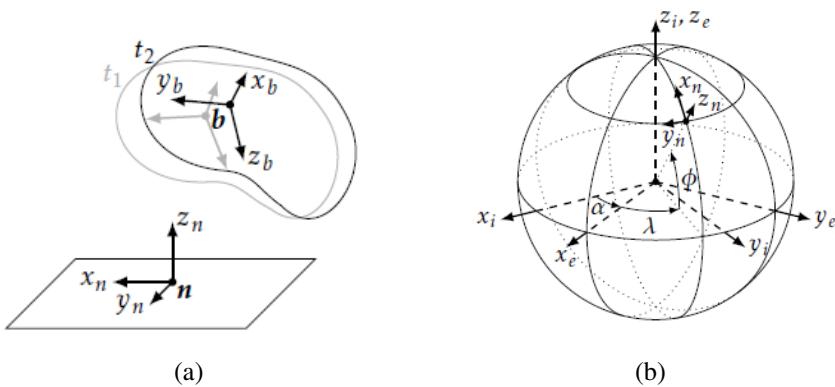


Figura 3.5: in 3.5(a) il *b-frame* nell'istante  $t_1$  e  $t_2$  relativamente al *n-frame*, in 3.5(b) l'*n-frame* in latitudine  $\varphi$  e longitudine  $\lambda$ , l'*e-frame* all'angolo  $\alpha(t) = \omega_{ie}t$  e l'*i-frame*

Ignorando la dipendenza dal tempo delle quantità coinvolte, la misura del giroscopio (si veda 3.2) è modellata in [17] come:

$$y_\omega = \omega_{ib}^b + \delta_\omega^b + e_\omega^b \quad (3.7)$$

Dove:

- $\omega_{ib}$  è la velocità angolare nel *b-frame* osservata dall'*i-frame*
- $\delta_\omega$  è la deriva del sensore che varia lentamente nel tempo
- $e_w^b$  è il rumore gaussiano

La velocità angolare  $\omega_{ib}$  può essere così estesa:

$$\omega_{ib} = R^{bn}(\omega_{ie}^n + \omega_{en}^n) + \omega_{nb}^b \quad (3.8)$$

Dove:

- $R$  è la matrice di rotazione
- $\omega_{ie}$  è la velocità angolare della terra
- $\omega_{en}$  è velocità angolare di trasporto
- $\omega_{nb}$  è la velocità angolare richiesta ai fini della navigazione

La misura dell'accelerometro  $y_a$  è invece modellata in [17] come:

$$y_a = f^b + \delta_a^b + e_a^b = R^{bn}(\ddot{b}_{ii}^n - g^n) + \delta_a^b + e_a^b \quad (3.9)$$

Dove:

- $f$  è la specifica forza esterna
- $\delta_a$  è la deriva del sensore che varia lentamente nel tempo
- $e_a$  è il rumore gaussiano

L'Eq.3.9 divide la forza specifica nei suoi contributi provenienti dall'accelerazione lineare del corpo osservata dall'*i-frame* ( $\ddot{b}_{ii}$ ) e dal vettore gravitazione  $g$ . L'accelerazione lineare può a sua volta essere espansa come:

$$\ddot{b}_{ii} = \omega_{ie}^n \times \omega_{ie}^n \times R^{ni} b^i + 2\omega_{ie}^n \times \dot{b}_n^n + \ddot{b}_{nn}^n \quad (3.10)$$

dove  $\ddot{b}_{nn}^n$  è l'accelerazione del corpo osservata dal *n-frame* richiesto per la navigazione.

Infine per il magnetometro (si veda 3.3) la misura  $y_m$  è così modellata:

$$y_m = m^b + e_b^b = R^{bn}m^n + e_m^b \quad (3.11)$$

Dove:

- $m$  è il vettore del campo magnetico locale
- $e_m$  è il rumore gaussiano

In assenza di oggetti ferromagnetici,  $m$  è il campo magnetico della terra e la misura del magnetometro può essere usata come una bussola per trovare la direzione del nord magnetico.

# Capitolo 4

## Stima dell'assetto tramite sensor fusion

In questo capitolo vengono inizialmente illustrati gli strumenti matematici utilizzati per rappresentare l'assetto di un corpo rigido nello spazio, successivamente viene illustrato l'algoritmo di fusione dei dati, provenienti dall'unità di misura inerziale, utilizzato per la stima dell'assetto dell'operatore.

### 4.1 Rappresentazione geometrica dell'assetto di un corpo rigido nello spazio

Con "*assetto di un corpo rigido*" si intende l'orientamento di un corpo rigido rispetto ad un particolare sistema di riferimento.

Tale orientamento è rappresentato da una matrice di rotazione che, applicata ad un qualsiasi vettore nel sistema di riferimento mobile, ne fornisce una rappresentazione nel sistema di riferimento fisso (3.4). Tale rotazione può essere espressa attraverso numerosi strumenti matematici, tra i più utilizzati si hanno:

- **Angoli di Eulero**
- **Quaternioni unitari**

Nella Tab.4.1 vengono riportate sinteticamente le caratteristiche delle rappresentazioni appena enunciate [20]:

Rappresentazione	#Parametri	Caratteristiche
Angoli di Eulero	3	<ul style="list-style-type: none"> <li>- facilmente interpretabili dall’essere umano</li> <li>- funzioni trigonometriche nelle relazioni cinematiche</li> <li>- soffrono del fenomeno noto come <i>Gimbal lock</i></li> <li>- meno accurati dei quaternioni</li> </ul>
Quaternioni unitari	4	<ul style="list-style-type: none"> <li>- non interpretabili facilmente dall’essere umano</li> <li>- equazioni della cinematica lineari</li> <li>- costo computazione di elaborazione minore degli angoli di Eulero</li> <li>- necessitano di un vincolo di norma unitaria</li> </ul>

Tabella 4.1: Tabella comparativa delle rappresentazioni d’assetto

Nell’algoritmo di fusione dei dati, dettagliato nei paragrafi successivi, si è adottato un approccio ibrido molto comune nei contesti applicativi dei sistemi IPS(1.2). Tale approccio consiste nell’utilizzare la rappresentazione mediante *quaternioni* per le computazioni mentre la rappresentazione mediante gli *angoli di Eulero* per la visualizzazione.

### 4.1.1 Matrice di rotazione

Di seguito si fornisce una definizione generale di matrice di assetto[21]. Si supponga di avere due sistemi di riferimento cartesiani in tre dimensioni  $F$  e  $G$ , una matrice ortogonale  $A_{FG}$ , detta di rotazione ed un vettore  $x_G$  espresso nel sistema di riferimento  $G$ .

La matrice  $A_{FG}$  permette di esprimere il vettore  $x_G$  rispetto al sistema di riferimento  $F$  secondo la seguente equazione:

$$x_F = A_{FG}x_G \quad (4.1)$$

Essendo per ipotesi la matrice  $A_{FG}$  ortogonale, l’operazione di inversione corrisponde al calcolo della sua trasposta:

$$x_G = A_{FG}^T x_F \quad (4.2)$$

Quindi determinare l’assetto significa definire la matrice di rotazione che permette, attraverso una semplice moltiplicazione, di ruotare i vettori da un sistema di riferimento mobile ad uno fisso.

### 4.1.2 Angoli di Eulero

Si consideri [21] un sistema di riferimento cartesiano  $F$  fisso (con assi  $x_F, y_F$  e  $z_F$ ) ed un sistema di riferimento cartesiano  $G$  mobile (con assi  $x_G, y_G$  e  $z_G$ ).

Affinché l’orientamento degli assi del sistema mobile  $G$  coincida con quelli del sistema fisso  $F$ , si devono eseguire almeno tre rotazioni successive attorno ai tre assi.

Tale vincolo è posto dal teorema di Eulero che è alla base di tutte le matrici di rotazioni. Il teorema afferma che:

- Per ogni rotazione, esiste sempre un vettore che avrà la medesima rappresentazione nei due sistemi di riferimento
- ogni rotazione avviene sempre attorno ad un asse fisso

L’idea è quella di ruotare ogni volta il sistema attorno ad uno dei suoi tre assi, così facendo l’asse attorno al quale è avvenuta la rotazione rimane fisso, mentre gli altri due cambiano orientamento. La rotazione successiva verrà fatta attorno ad uno dei

due precedenti assi che hanno mutato l’orientamento. Con sistemi cartesiani a tre assi è possibile quindi scegliere tra dodici differenti sequenze di rotazioni per un totale eguale di possibili rappresentazioni della matrice di rotazione.

Nell’ambito di questa tesi e più comunemente in quello aeronautico, si è utilizzata la sequenza di rotazioni *z-y-x* e gli angoli  $\psi, \vartheta$  e  $\varphi$  chiamati rispettivamente *imbardata, beccheggio e rollio* (in inglese *yaw, pitch e roll*) mostrati in Fig.4.1:

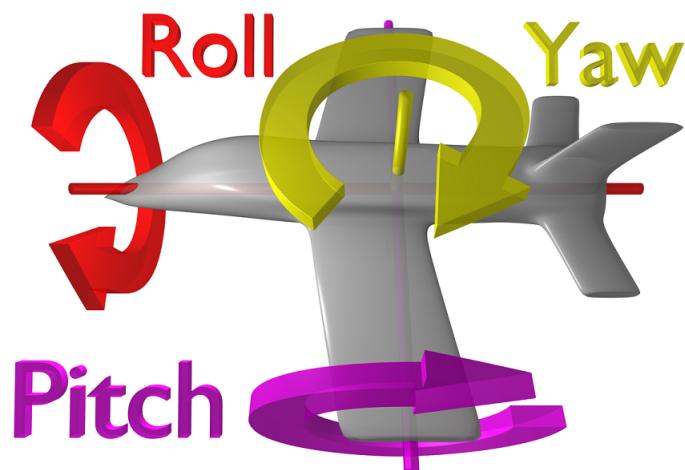


Figura 4.1: Rappresentazione degli angoli di roll, pitch e yaw per un velivolo [22]

Le tre rotazioni in questione sono:

$$A(z, \psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

$$A(y, \vartheta) = \begin{bmatrix} \cos \vartheta & 0 & \sin \vartheta \\ 0 & 1 & 0 \\ -\sin \vartheta & 0 & \cos \vartheta \end{bmatrix} \quad (4.4)$$

$$A(x, \varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix} \quad (4.5)$$

Andando a moltiplicare le precedenti matrici si ottiene la matrice di rotazione cercata:

$$A(\psi, \vartheta, \varphi) = \begin{bmatrix} \cos \psi \cos \vartheta & \cos \psi \sin \vartheta \sin \varphi - \sin \psi \cos \varphi & \cos \psi \sin \vartheta \cos \varphi + \sin \psi \sin \varphi \\ \sin \psi \cos \vartheta & \sin \psi \sin \vartheta \sin \varphi + \cos \psi \cos \varphi & \sin \psi \sin \vartheta \cos \varphi - \cos \psi \sin \varphi \\ -\sin \vartheta & \cos \vartheta \sin \varphi & \cos \vartheta \cos \varphi \end{bmatrix} \quad (4.6)$$

Il significato geometrico si ha osservando la Fig.4.2, dove:

- la linea dei nodi è definita come l’intersezione tra il piano individuato dagli assi  $x_Fy_F$  e quello individuato dagli assi  $y_Bz_B$
- $\psi$  è l’angolo tra  $y_F$  e la linea dei nodi
- $\vartheta$  è l’angolo tra  $x_B$  e la sua proiezione sul piano  $x_Fy_F$
- $\varphi$  è l’angolo tra  $y_B$  e la linea dei nodi

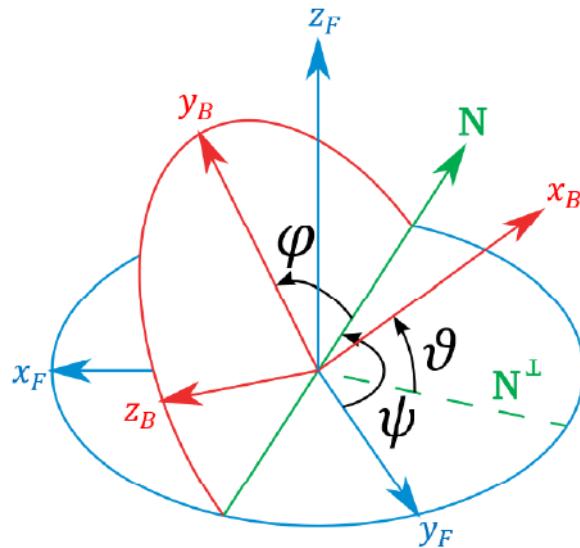


Figura 4.2: Significato geometrico degli angoli di Eulero [21]

Come accennato nella tabella 4.1, la rappresentazione dell’assetto mediante angoli di Eulero presenta un problema di singolarità noto come *gimbal lock*.

Nel caso della matrice di rotazione ricavata nell’equazione 4.6, il problema si presenta per  $\vartheta = \pm \frac{\pi}{2}$ : in questa situazione esistono infinite combinazioni di  $\varphi$  e  $\psi$  che

portano alla stessa matrice di rotazione, nel caso di  $\vartheta = \frac{\pi}{2}$  si ha:

$$A(\psi, \vartheta, \varphi) = \begin{bmatrix} 0 & \sin(\varphi - \psi) & \cos(\varphi - \psi) \\ 0 & \cos(\varphi - \psi) & -\sin(\varphi - \psi) \\ -1 & 0 & 0 \end{bmatrix} \quad (4.7)$$

Ottenuta mediante sostituzione e applicazione delle formule di addizione e sottrazione di seno e coseno. Mentre la matrice di rotazione in Eq.4.6 permette una rotazione completa attorno ad un qualsiasi asse, la matrice in 4.7 permette la rotazione attorno al solo asse  $X$ . Quindi si manifesta un vincolo di rotazione con conseguente perdita di un grado di libertà, come mostrato in Fig.4.3.

Basti pensare ad un velivolo con un *pitch* di 90, modificare il *roll* o lo *yaw* di un certo angolo avrebbe il medesimo effetto.

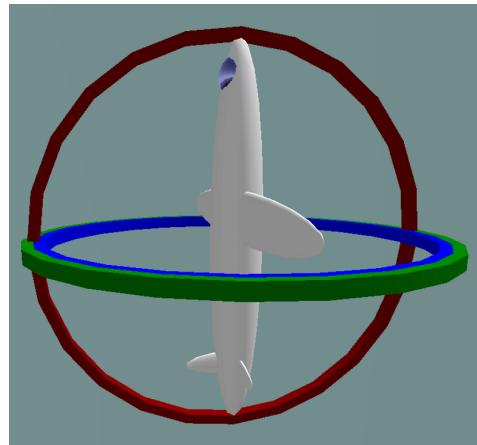


Figura 4.3: Rappresentazione del fenomeno di *gimbal lock* [23]

### 4.1.3 Quaternioni unitari

I quaternioni hanno la peculiarità di essere il metodo di rappresentazione dell’assetto con minor numero di parametri privi di singolarità, come ad esempio il *gimbal lock* visto precedentemente per gli angoli di Eulero.

Il quaternione unitario è un vettore composto da tre elementi che definiscono il vettore  $q_{1:3}$  e da un elemento scalare  $q_4$ , tale che la norma:

$$\|\vec{q}\| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} = 1 \quad (4.8)$$

Questi possono essere usati per rappresentare l’assetto di un corpo rigido in quanto le trasformazioni che legano le terne di Eulero ai quaternioni unitari, sono semplicemente delle trasformazioni algebriche che portano da uno spazio rappresentativo all’altro.

Sfruttando il teorema di Eulero [21], si può parametrizzare la matrice di rotazione in 4.6 ottenendo l’equivalente in funzione dei quaternioni:

$$A(\vec{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 - q_4^2 & 2(q_1 q_2 + q_3 q_4) & 2(q_1 q_3 - q_2 q_4) \\ 2(q_2 q_1 - q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 + q_1 q_4) \\ 2(q_3 q_1 + q_2 q_4) & 2(q_3 q_2 - q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \quad (4.9)$$

Come si può notare dalla 4.6, i quaternioni permettono di definire una matrice di assetto i cui elementi sono funzioni quadratiche omogenee degli elementi del quaternione. Si evita così qualsiasi tipo di calcolo trigonometrico (e relative singolarità) e si ha un costo computazionale minore.

Un diretto confronto tra le matrici 4.6 e 4.9 permette inoltre di definire il rapporto tra gli angoli di Eulero ( $\psi, \vartheta, \varphi$ ) e le componenti del quaternione ( $q_1, q_2, q_3, q_4$ ) attraverso le seguenti:

$$\psi = \text{Atan2}\left(\frac{2q_2 q_3 - 2q_1 q_4}{2q_1^2 + q_2^2 - 1}\right) \quad (4.10)$$

$$\vartheta = -\sin(2q_2 q_4 + 2q_1 q_3) \quad (4.11)$$

$$\varphi = \text{Atan2}\left(\frac{2q_3 q_4 - 2q_1 q_2}{2q_1^2 + q_4^2 - 1}\right) \quad (4.12)$$

## 4.2 Algoritmo di sensor fusion per la stima dell’assetto

La maggior parte degli algoritmi di stima dell’assetto che sfruttano i dati provenienti da sensori (sez.3) si basano sull’applicazione di un *filtro di Kalman*.

Il filtro di Kalman è un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico a partire da una serie di misure soggette a rumore. Per le sue caratteristiche intrinseche è un filtro ottimo per rumori e disturbi agenti su sistemi gaussiani a media nulla [26], per maggiori dettagli si rimanda alla lettura dell’Appendice B.

Con riferimento alla rappresentazione del sistema realizzato nell’ambito di questa tesi (sez.2.3), all’interno del ”modulo” *microcontroller* e del ”modulo” *App* vi sono due diversi algoritmi di *sensor fusion* basati sul filtro di Kalman che, a partire dai dati grezzi letti dall’unità di misura inerziale (sez.3), stimano l’assetto relativo al modello di misura (sez.3.4).

Il modulo *Microcontroller* utilizza una libreria esterna realizzata da *STM* [25] della quale però non vengono forniti i dettagli implementativi, perciò nei paragrafi successivi si farà riferimento all’algoritmo utilizzato all’interno del modulo *App* che è l’implementazione di un lavoro di tesi trovato in letteratura [24].

#### 4.2.1 Equazioni di stato del sistema

E’ necessario come prima cosa, descrive l’evoluzione del sistema con un’equazione differenziale, dove lo stato del sistema è rappresentato tramite quaternioni unitari (4.1.3).

Definiamo con  $\omega_{nb}^n$  il vettore delle velocità angolari rispetto al *b-frame* dell’IMU (sez.3.4):

$$\omega_{nb}^b = [\omega_x, \omega_y, \omega_z]^T \quad (4.13)$$

Dall’algebra dei quaternioni è possibile scrivere l’equazione che descrive l’evoluzione del sistema come:

$$\dot{q}_n^b = \frac{1}{2} \Omega_{nb}^n q_n^b \quad (4.14)$$

Dove  $\Omega_{nb}^n$  è la matrice con gli elementi del vettore della velocità angolare in 4.13:

$$\Omega_{nb}^n = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \quad (4.15)$$

Si noti bene come dalla precedente equazione 4.14 l’evoluzione del sistema, corrispondente alla fase detta ”aggiornamento a priori” del filtro di Kalman, è caratterizzata dai soli dati del giroscopio. Come si vedrà tra poco, i sensori di accelerometro (sez.3.1) e magnetometro (sez.3.3) verranno utilizzati nella fase detta ”aggiornamento a posteriori” per correggere la stima ottenuta con il solo giroscopio.

### 4.2.2 Filtro di Kalman a due stadi

Un filtro di Kalman completo potrebbe essere computazionalmente troppo oneroso da eseguire per un dispositivo come un microcontrollore. Per questo motivo in [24] si è pensato di adottare un approccio diverso dall’implementazione classica del filtro. Tale metodo è ancora basato sul filtro di Kalman, ma la fase di correzione della stima ottenuta nell’aggiornamento a priori, viene divisa in due parti, ognuna delle quali agisce su una specifica componente.

Il primo stadio utilizza i dati provenienti dall’accelerometro (sez.3.1) per correggere gli angoli stimati di *roll* e *pitch* (sez.4.1.2) mentre il secondo stadio utilizza i dati provenienti dal magnetometro (sez.3.3) per correggere la stima dello *yaw*. Ogni stadio lavora con un singolo vettore di dati grezzi che è usato appunto per correggere una specifica parte della stima dell’assetto, da questo ne consegue che ogni stadio può lavorare con una matrice di dimensione  $4 \times 3$  contro le  $4 \times 6$  di un filtro di Kalman completo. Da questo ne consegue un guadagno notevole in termini di prestazioni.

Nella figura seguente una rappresentazione a blocchi del filtro a due stadi:

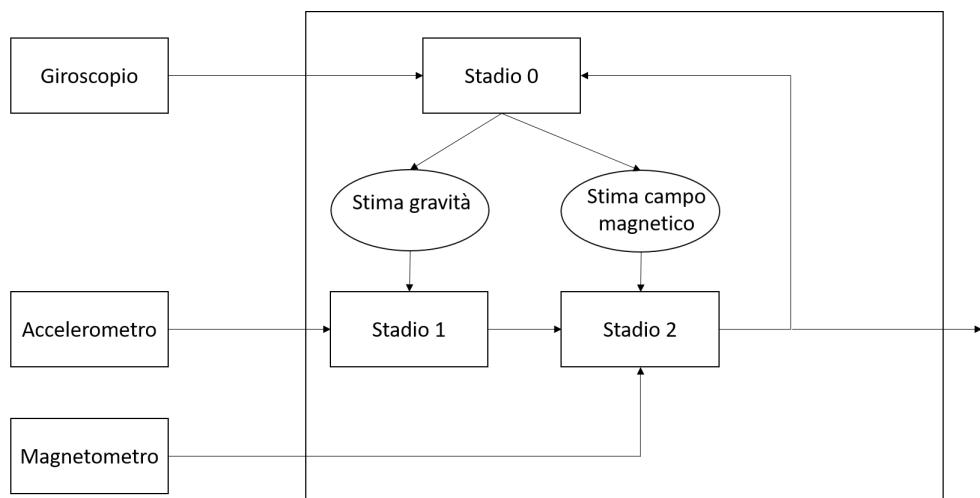


Figura 4.4: Rappresentazione del principio di funzionamento del filtro di Kalman a due stadi [24]

Un altro grande vantaggio consiste nella flessibilità dell’algoritmo, questo infatti può essere utilizzato anche se si dispone di un IMU a 6DOF (privo del magnetometro) semplicemente ”disattivando” lo *stadio 2* in Fig.4.4, così facendo l’uscita dallo *stadio 1* diventa l’uscita del sistema e viene retroazionata allo *stadio 0* andando a correggere la prossima lettura.

### 4.2.3 Descrizione dell’algoritmo

Lo stato del sistema è, come già detto, rappresentato tramite quaternioni (sez.4.1.3). Lo *stadio 0* anche detto *stima a priori* utilizza solo i dati grezzi provenienti dal giroscopio. Fatto ciò, si deve stimare il vettore gravitazionale  $h_1$  per poter correggere il *roll* e il *pitch* nello *stadio 1*. Tale vettore è così definito:

$$h_1(q_k) = \hat{q} = R_n^b \begin{bmatrix} 0 \\ 0 \\ |q| \end{bmatrix} = |g| \begin{bmatrix} 2q_2q_4 - 2q_1q_3 \\ 2q_1q_2 + 2q_3q_4 \\ q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \quad (4.16)$$

Il vettore gravitazionale  $g$ , espresso secondo l’*i-frame* (sez.3.4), viene riportato nel *b-frame* utilizzando la seguente matrice di rotazione:

$$R_n^b = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2q_3 + q_1q_4) & 2(q_2q_4 - q_1q_3) \\ 2(q_2q_3 - q_1q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3q_4 + q_1q_2) \\ 2(q_2q_4 + q_1q_3) & 2(q_3q_4 - q_1q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix} \quad (4.17)$$

Da  $h_1$  si ricava quindi la matrice  $H_{k1}$  del guadagno di Kalman che verrà utilizzata successivamente dallo *stadio 1*:

$$H_{k1} = \frac{\partial h_{1[i]}}{\partial q_j} = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 2q_1 & -2q_2 & -2q_3 & 2q_4 \end{bmatrix} \quad (4.18)$$

Lo *stadio 1* calcola il residuo  $q_{\epsilon 1}$  che sarà sommato allo stato  $\hat{x}_k$ , stimato a *priori* dallo *stadio 0*, ottenendo così la prima stima a *posteriori*  $\hat{x}_{k1}$ . L’elemento  $q_{\epsilon 1,4}$  è posto a zero, al fine di essere sicuri che lo *stadio 1* corregga solo *roll* e *pitch*. Il residuo  $q_{\epsilon 1}$  è determinato dalla seguente relazione:

$$q_{\epsilon 1} = K_{k1}(z_{k1} - h_1(\hat{x}_k, 0)) = q_{\epsilon 1,1} + q_{\epsilon 1,2} + q_{\epsilon 1,3} + 0 \cdot q_{\epsilon 1,4} \quad (4.19)$$

e quindi la stima dello stato a *posteriori* risulta:

$$\hat{x}_{k1} = \hat{x}_k + q_{\epsilon 1} \quad (4.20)$$

Per lo *stadio 2* è necessario stimare il vettore del campo magnetico  $h_2$ :

$$h_2(q_k) = \hat{m} = R_n^b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2q_2q_3 + 2q_1q_4 \\ q_1^2 - q_2^2 - q_3^2 - q_4^2 \\ 2q_3q_4 - 2q_1q_2 \end{bmatrix} \quad (4.21)$$

La matrice  $H_{k2}$  usata dallo *stadio 2* per calcolare il guadagno di Kalman è la seguente:

$$H_{k2} = \frac{\partial h_{2[i]}}{\partial q_j} = \begin{bmatrix} 2q_4 & 2q_3 & 2q_2 & 2q_1 \\ 2q_1 & -2q_2 & -2q_3 & -2q_4 \\ -2q_2 & -2q_1 & -2q_4 & 2q_3 \end{bmatrix} \quad (4.22)$$

Si ottiene così il secondo residuo  $q_{\epsilon 2}$  che verrà sommato alla precedente stima a *posteriori* determinata dallo *stadio 1*. In questo caso, al fine di essere sicuri che lo *stadio 2* corregga solo lo *yaw*, le componenti del residuo poste a zero sono  $q_{\epsilon 2,2}$  e  $q_{\epsilon 2,3}$ :

$$q_{\epsilon 2} = q_{\epsilon 2,1} + 0 \cdot q_{\epsilon 2,2} + 0 \cdot q_{\epsilon 2,3} + q_{\epsilon 2,4} \quad (4.23)$$

E infine l’ultima stima a *posteriori* è data da:

$$\hat{x}_k = \hat{x}_{k1} + q_{\epsilon 2} \quad (4.24)$$

#### 4.2.4 Trasformazione del sistema a tempo discreto

L’equazione precedenti descrivono l’evoluzione del sistema nel dominio del tempo continuo. Per implementare tale filtro si devono trasformare l’equazioni in 4.2.3 nel dominio del tempo discreto.

In [24] si utilizzano le formule standard per discretizzare un sistema a tempo continuo in uno a tempo discreto. Il sistema di partenza è descritto da un’equazione del tipo:

$$\dot{x} = A_{TC}x(t) + B_{TC}u(t) \quad (4.25)$$

dove  $A_{TC}$  e  $B_{TC}$  sono le matrici di stato e d’ingresso mentre  $x(t)$  e  $u(t)$  sono le variabili di stato e dell’ingresso nel tempo continuo.

Dalla definizione di derivata si ottiene la seguente relazione:

$$\dot{x} = \lim_{T \rightarrow 0} \frac{x(t+T) - x(t)}{T} = A_{TC}x(t) + B_{TC}u(t) \quad (4.26)$$

La variabile  $T$  è il *delta-time* trascorso tra ogni iterazione in un sistema digitale, quindi è possibile rimuovere l’operazione di limite ed ottenere:

$$x(t+T) = x(t) + A_{TC}x(t)T + B_{TC}u(t)T = (I + A_{TC}T)x(t) + B_{TC}Tu(t) \quad (4.27)$$

In conclusione il sistema a tempo continuo di partenza nell’equazione 4.25 si trasforma nel seguente sistema a tempo discreto:

$$x_k = Ax_{k-1} + Bu_k \quad (4.28)$$

Dove  $x_k$  è equivalente alla variabile  $x(t)$  campionata all’istante  $t + T$  e:

$$A = I + A_{TC}T \quad (4.29)$$

$$B = B_{TC}T \quad (4.30)$$

#### 4.2.5 Riepilogo algoritmo step-by-step

Per maggiore chiarezza si fornisce un riepilogo dell’algoritmo step-by-step da eseguire ad ogni iterazione:

*Inizio*

- Calcolo della matrice di transizione dello stato discreta  $A_k = I + \frac{1}{2}\Omega_{nb}^n T$
- Calcolo della stima a *priori* dello stato mediante lo *stadio 0*,  $\hat{q}_k = A_k \hat{q}_{k-1}$
- Calcolo della matrice di covarianza del rumore  $P_k = A_k P_{k-1} A_k^T + Q_k$

*Inizio della correzione mediante lo stadio 1*

- Calcolo della matrice  $H_{k1} = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 2q_1 & -2q_2 & -2q_3 & 2q_4 \end{bmatrix}$
- Calcolo del guadagno di Kalman  $K_{k1} = P_k H_{k1}^T (H_{k1} P_k H_{k1}^T + V_{k1} R_{k1} V_{k1}^T)^{-1}$
- Calcolo di  $h_1(\hat{q}_k) = \begin{bmatrix} 2q_2 q_4 - 2q_1 q_3 \\ 2q_1 q_2 + 2q_3 q_4 \\ q_1^2 - q_2^2 - q_3^2 + q_4^3 \end{bmatrix}$
- Calcolo del fattore di correzione  $q_{\in 1} = K_{k1}(z_{k1} - h_1(\hat{q}_k))$  con  $q_{\in 1,4} = 0$ , dove  $z_{k1}$  corrisponde alle accelerazioni sui tre assi lette dall’accelerometro.
- Calcolo della prima stima dello stato a *posteriori*  $\hat{q}_{k1} = \hat{q}_k + q_{\in 1}$
- Calcolo della matrice di covarianza del rumore a *posteriori*  $P_{k1} = (I - K_{k1} H_{k1}) P_k$

*Inizio della correzione mediante lo stadio 2*

- Calcolo della matrice  $H_{k2} = \begin{bmatrix} 2q_4 & 2q_3 & 2q_2 & 2q_1 \\ 2q_1 & -2q_2 & -2q_3 & -2q_4 \\ -2q_2 & -2q_1 & -2q_4 & 2q_3 \end{bmatrix}$
- Calcolo del guadagno di Kalman  $K_{k2} = P_k H_{k2}^T (H_{k2} P_k H_{k2}^T + V_{k2} R_{k2} V_{k2}^T)^{-1}$
- Calcolo di  $h_2(\hat{q}_k) = \begin{bmatrix} 2q_2 q_3 + 2q_1 q_4 \\ q_1^2 - q_2^2 - q_3^2 - q_4^2 \\ 2q_3 q_4 - 2q_1 q_2 \end{bmatrix}$
- Calcolo del secondo fattore di correzione a *posteriori*  $q_{\in 2} = K_{k2}(z_{k2} - h_2(\hat{q}_k))$  con  $q_{\in 2,2}, q_{\in 2,3} = 0$ , dove  $z_{k2}$  corrisponde alle misure del campo magnetico sui tre assi lette dal magnetometro.
- Calcolo della seconda stima dello stato a *posteriori*  $\hat{q}_{k2} = \hat{q}_{k1} + q_{\in 2}$
- Calcolo della seconda matrice di covarianza del rumore  $P_k = (I - K_{k2} H_{k2}) P_{k1}$

*fine*

#### 4.2.6 Settaggio iniziale delle costanti

Di seguito si mostrano i valori assegnati alle costanti nell’algoritmo precedentemente illustrato (sez.4.2.5). Tali valori sono definiti in [24] e, a seguito di alcuni test, si sono rivelati i valori ottimali anche nel lavoro di questa tesi.

- $Q = 10^{-6} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- $R_1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$
- $R_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- $P_0 = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$
- $q_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$
- $V_{[i,j]} = \frac{\delta h_{[i]}}{\delta v_{[j]}}(x_k, 0) = I$ , il che vuol dire assumere che il rumore dell’accelerometro e del magnetometro sia indipendente dalla posizione dell’IMU.

# **Capitolo 5**

## **Implementazione**

In questo capitolo vengono illustrati i dettagli implementativi riguardanti le realizzazioni hardware e software del sistema realizzato nell’ambito di questa tesi.

### **5.1 Implementazione Hardware**

In questa sottosezione verrà mostrato inizialmente uno schema con i componenti hardware utilizzati per realizzare il sistema ideato in sez.2, per finire verranno illustrati i canali di comunicazione.

#### **5.1.1 Schema hardware del sistema**

Con riferimento alla rappresentazione astratta del sistema a moduli presentata nella sez.2.3, segue una il medesimo schema ad un livello meno astratto nel quale si mostrano le componenti hardware che costituiscono i due moduli App e microcontrollore.

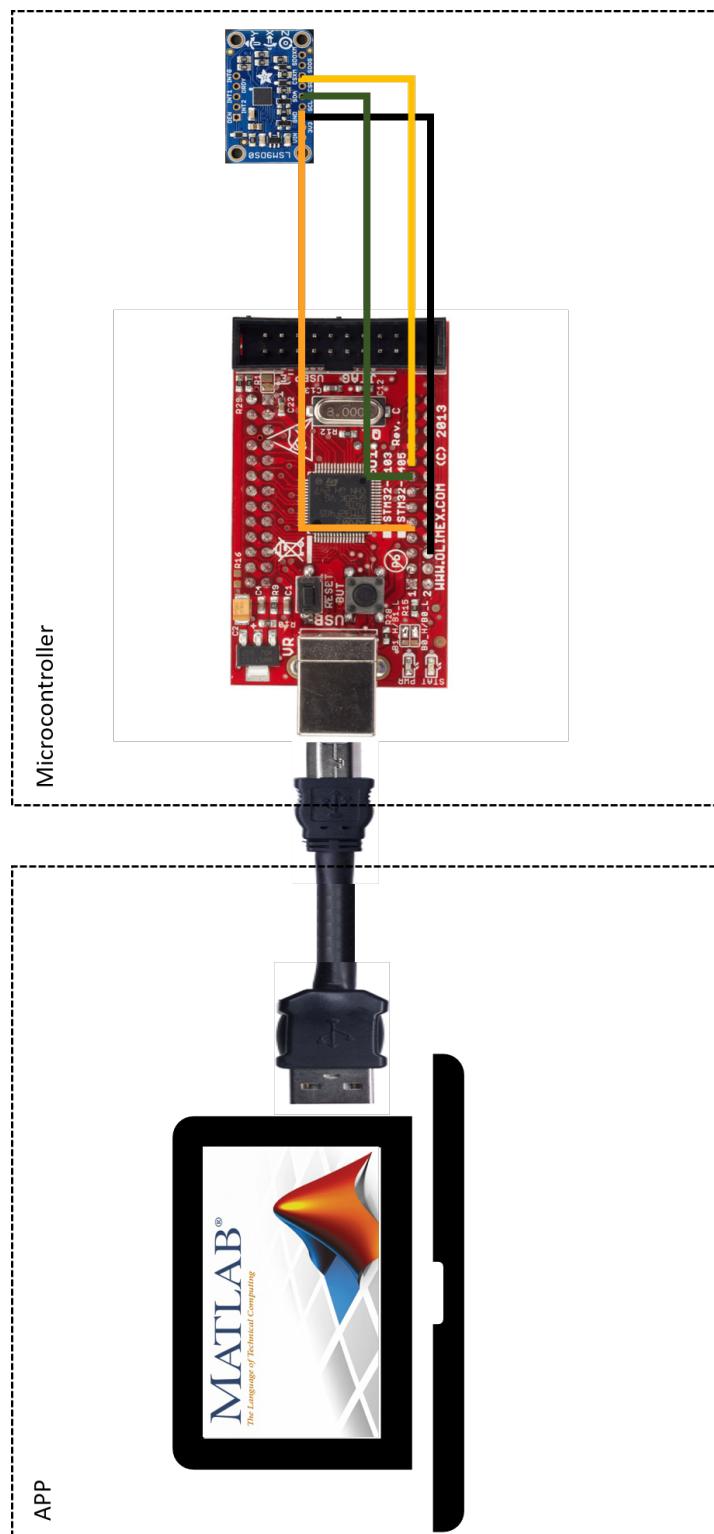


Figura 5.1: Schema hardware del sistema

Dove:

- **Il modulo APP:** è composto da un’elaboratore nel quale si utilizzano i seguenti software
  - *RealTerm* per catturare in un file di log i dati inviati dal moduli microcontrollore
  - *Matlab* per testare l’algoritmo di *Sensor Fusion* ed effettuare le varie analisi (sez.6)
- **Il modulo Microcontrollore:** è composto da
  - Una board Olimex con microcontrollore STM32F405 prodotto da *STM*
  - Un’unità di misura inerziale LSM9DS1 prodotta da *STM*

E’ necessario spendere due parole a riguardo dell’implementazione del modulo APP mostrata in Fig.5.1.

Relativamente al contesto applicativo individuato per questo sistema (sez.1) l’applicativo Matlab e, più in generale, l’elaboratore non sono adatti alla realizzazione di un sistema IPS real-time. Tuttavia nel lavoro di questa tesi, questi vengono utilizzati al solo fine di comparare l’algoritmo di *Sensor fusion* implementato (sez.4.2) con quello eseguito all’interno del microcontrollore e di effettuare ulteriori analisi (sez.6). Le fasi successive dello sviluppo di questo sistema, prevedono infatti lo studio della realizzazione ottima del modulo APP a partire dai risultati ottenuti dal lavoro di questa tesi.

### 5.1.2 Canali di comunicazione

In questa sezione verranno illustrati i canali di comunicazione (mostrati in Fig.5.1) utilizzati per la trasmissione dei dati all’interno del sistema realizzato nell’ambito di questa tesi. Il primo paragrafo tratta il canale di comunicazione utilizzato tra il microcontrollore e l’unità di misura inerziale mentre nel secondo e ultimo paragrafo il canale di comunicazione utilizzato per trasmettere dati dal microcontrollore all’applicativo terminale sull’elaboratore.

### 5.1.2.1 I2C

Per la comunicazione tra l' **IMU** e il **Microcontrollore**, si è utilizzato un canale di comunicazione seriale bifilare noto come *Inter Integrated Circuit*, abbreviato **I2C**, come mostrato dalla seguente figura:

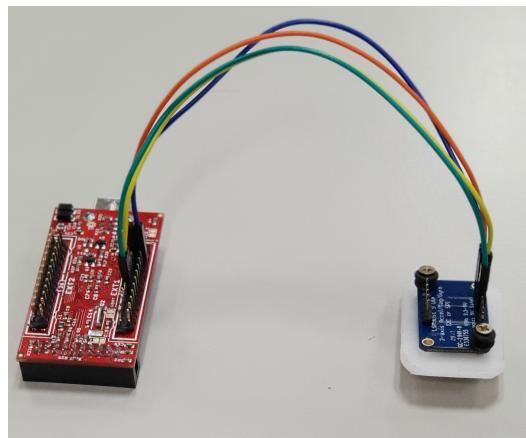


Figura 5.2: Foto del canale di comunicazione I2C tra il microcontrollore e l'unità di misura inerziale.

Il protocollo [27] dell'I2C richiede due linee seriali per comunicare correttamente:

- SDA (*Serial Data*) per i dati
- SCL (*Serial Clock*) per sincronizzare i dispositivi

Vanno inoltre aggiunte una connessione di riferimento alla massa e una alla linea di alimentazione (tipicamente 5 o 3,3 V) a cui sono connessi resistori di *pull-up*, come mostrato dalla figura seguente:

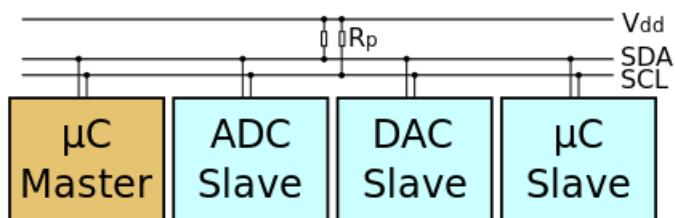


Figura 5.3: Rappresentazione di dispositivi collegati mediante I2C

L’I2C utilizza un indirizzo a 7 bit per un totale di 128 indirizzi disponibili. Poiché di questi 16 sono riservati, si ha un totale finale di 112 dispositivi collegabili alla medesima linea.

La velocità è il grande limite di questa comunicazione, infatti con le recenti revisioni si è riuscita a raggiungere una velocità massima di 3,4 Mbit/s (detto anche *High Speed Mode*). Nello specifico di questa tesi lo si è utilizzato in *fast mode* che corrisponde ad una velocità di 400 Kbit/s.

I dispositivi collegati al bus possono essere di due tipi:

- Un *Master* che controlla la SCL, inizializza le comunicazioni e invia dati
- Uno o più *Slave* che rispondono ad un master e inviano dati

Il messaggio viene spezzato in due parti:

- *Address frame* dove il *master* indica l’indirizzo dello *slave* interessato alla comunicazione
- uno o più *Data frames* contenenti l’informazione da trasmettere

I dati sono ”piazzati” sulla linea SDA dopo che la linea SCL è posta a zero dal master, questi dati verranno letti dal dispositivo specificato nell’*address frame* quando il *master* riporterà la linea SCL al valore alto.

Relativamente al lavoro di questa tesi si identificano il *Microcontrollore* come *master* mentre l’*IMU* come *slave*. Nello specifico la comunicazione consiste nella lettura periodica da parte del microcontrollore dei valori misurati dai sensori all’interno dell’IMU. Tale lettura è stata realizzata leggendo i singoli registri ad 8 bit dei sensori (per maggiori dettagli si veda la stima temporale in sez.6.1) attraverso lo script in Appendice A.1.1 e rappresentato dalla seguente figura:

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

Figura 5.4: Rappresentazione del flusso di dati generato per la lettura, tramite I2C, di un registro ad 8 bit.

Dove:

- *ST* è il segnale di inizio di una comunicazione

- $SAD + W$  include l'indirizzo del dispositivo slave mentre l'ultimo bit specifica che il master sta per scrivere
- $SAK$  lo slave con l'indirizzo specificato precedentemente risponde con un segnale di *acknowledge*
- $SUB$  è l'indirizzo del registro interno del dispositivo specificato precedentemente
- $SAK$  lo slave risponde con un segnali di *acknowledge* se l'indirizzo del registro esiste
- $SR$  è il bit di ripetizione dello start utilizzato in combinazione con  $SAD+R/W$
- $SAD + R$  include l'indirizzo del dispositivo slave mentre l'ultimo bit specifica che il master vuole ricevere dati
- $SAK$  segnale di *acknowledge* da parte dello *slave*
- $DATA$  dati trasmitti nel formato ad 8 bit
- $NMAK$  segnale di *non acknowledge* da parte del *master*
- $SP$  bit di stop della comunicazione

Tuttavia questa lettura non è sempre la migliore, infatti è possibile continuare a leggere i registri interni dello *slave* trasmettendo il segnale di *MAK* invece del segnale di *NMAK* mostrato in Fig.5.4. In questo modo l'IMU incrementerà l'indirizzo del registro interno di partenza specificato in  $SUB$  e provvederà a trasmettere il valore del registro interno successivo ad esso. Questa funzionalità è molto utile quando si devono leggere dati, come in questo caso, che sono rappresentati con più registri. Così facendo infatti si riduce l'*overhead* della trasmissione e si migliora quindi il goodput generale rispetto alla lettura ripetuta dei singoli registri.

### 5.1.2.2 USB CDC

Per la comunicazione tra il *microcontrollore* e l'elaboratore si è utilizzato un canale di comunicazione USB nella classe CDC [29], mostrato in Fig.??, attraverso la libreria *HAL* [28] fornita da *STM*.

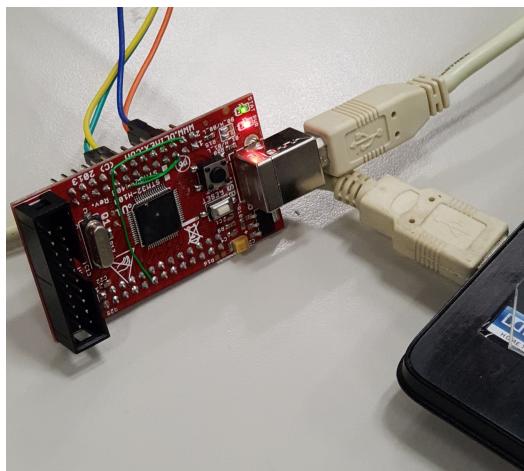


Figura 5.5: Foto del canale di comunicazione USB usato nella classe CDC per la trasmissione dei dati dal microcontrollore all’elaboratore.

Un dispositivo CDC è composto [30] dalle seguenti interfacce:

- *Communications Class Interface* (CCI)
- *Data Class Interface* (DCI)

L’interfaccia CCI è responsabile della gestione del dispositivo e in alcuni casi anche della gestione delle chiamate. Per gestione del dispositivo si intendono le configurazioni necessarie da eseguire sul *device* e la notifica degli eventi, mentre per gestione della chiamata si intendono le operazioni necessarie per stabilire e terminare una comunicazione. La CCI è obbligatoria per tutti i dispositivi CDC in quanto lo identifica specificando il modello di comunicazione supportato dal dispositivo stesso.

L’interfaccia DCI è responsabile della trasmissione dei dati, infatti i dati trasmessi e/o ricevuti non seguono uno specifico formato. Tutte le interfacce di questo tipo possono essere viste come interfacce subordinate della CCI.

Tutti i dispositivi CDC devono avere almeno un’interfaccia CCI e zero o più interfacce DCI. Un CCI e ogni suo DCI subordinato forniscono una specifica funzionalità al dispositivo.

In generale un *device* CDC potrebbe supportare diverse funzionalità e quindi essere composto da diversi set di CCI e DCI, come mostrato dalla figura seguente:

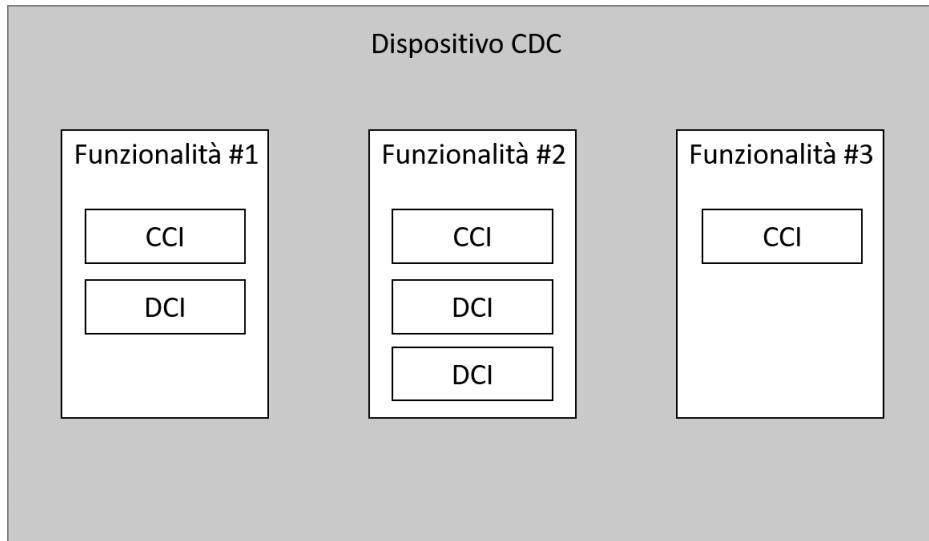


Figura 5.6: Rappresentazione di un dispositivo CDC che supporta tre diverse funzionalità.

Un dispositivo CDC solitamente usa le seguenti combinazioni di *endpoints*:

- Una coppia di *endpoints* di controllo IN e OUT chiamate *endpoint di default*.
- Un *endpoint* opzionale chiamato *bulk* o *interrupt IN*
- Una coppia di *bulk* o *endpoints* IN e OUT asincroni

La tabella 5.1.2.2 mostra gli usi dei differenti *endpoints* sopracitati e da quale interfaccia sono utilizzati.

Endpoint	Direzione	Interfaccia	Uso
Control IN	Device->Host	CCI	Richiesta standard per l'enumerazione, richiesta specifica della classe, controllo del dispositivo e controllo della chiamata
Control OUT	Host->Device	CCI	”...”
Interrupt o bulk IN	Device->Host	CCI	Notifica di un evento come lo stato della linea o della rete
Bulk o OUT asincrono	Host->Device	DCI	Comunicazione di dati grezzi o formattati
Bulk o IN asincrono	Host->Device	DCI	”...”

Tabella 5.1: Tabella con gli usi tipici degli *endpoints* per comunicazioni USB-CDC

La libreria HAL utilizzata nel lavoro di questa tesi, utilizza due tipi di *endpoints* per la comunicazione:

- *Endpoint Bulk* per il trasferimento dei dati (1 *endpoint* di OUT e 1 *endpoint* di IN)
- *Endpoint interrupt* per il controllo della comunicazione (richieste CDC, 1 *endpoint* IN)

Il trasferimento dei dati IN (dal *device* verso l'*host*) è gestito periodicamente in base alle richieste dell'*host* (il *device* specifica l'intervallo di tempo tra le richieste dei pacchetti). Per questo motivo, un buffer statico circolare viene usato per memorizzare i dati mandati dal terminale del *device* (esempio USART nel caso di una *Virtual COM Port*).

In generale, la comunicazione USB è molto più veloce dell'output del terminale (ad esempio il bitrate massimo di un USART è tipicamente 115 Kbps mentre il bitrate dell'USB 1.1 è 12 Mbps in *Full speed mode* o 480 Mbps in *High speed mode*). Di conseguenza, prima di mandare nuovi pacchetti l'*host* deve aspettare finché il *device* non termina di processare i dati precedentemente inviati. Quindi, nel caso di data OUT dall'*host* verso il *device* l'uso di un buffer statico circolare non avrebbe senso. Il driver sul *device* aspetta che la procedura di processamento del dato precedente

sia terminata prima di mandare un segnale di *acknowledge* all’*host*.

Infine la gestione delle richieste di controllo viene effettuata utilizzando o un *endpoint* pari a zero oppure attraverso un *endpoint interrupt* infatti se la dimensione del pacchetto di richiesta non eccede i 64 bytes, allora l’*endpoint 0* è sufficiente a gestire tale richiesta.

## 5.2 Implementazione Software

In questa sezione si mostrano alcuni dettagli riguardanti l’implementazione software. Nella parte iniziale si fornisce un *overview* delle funzionalità dei due moduli (sez.2.3) mentre nell’ultima parte si mostrano le modalità operative identificate e realizzate.

### 5.2.1 Diagramma Comportamentale

Per una avere una visione generale delle funzionalità implementate nei moduli APP e microcontrollore, si mostrano due pseudo diagrammi di flusso che ne sintetizzano il comportamento di ciascuno dei due.

#### 5.2.1.1 Diagramma del Microcontrollore

Il comportamento del modulo Microcontrollore può essere rappresentato mediante il seguente diagramma di flusso:

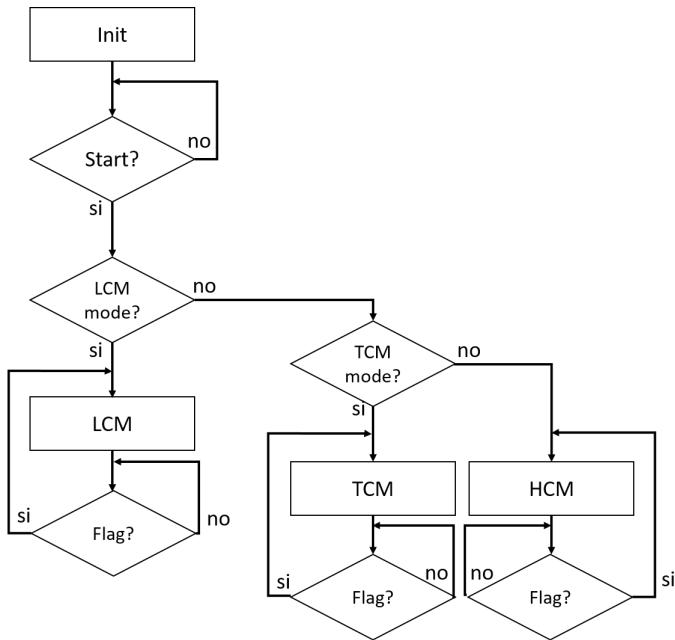


Figura 5.7: Rappresentazione del comportamento per il modulo Microcontrollore.

Dove:

- **Init**: rappresenta la fase di inizializzazione del micro. L’utente sceglie la *Computation Mode* e il numero di sensori da utilizzare (6DOF o 9DOF) modificando semplicemente due variabili globali (si veda appendice A.1.3).
- **Start?**: il microcontrollore attende che il pulsante, integrato nella board, venga premuto di iniziare qualsiasi operazione.
- **LCM mode? e TCM mode?**: test di condizione sulla variabile globale che indica la *Computation Mode* settata nella fase di inizializzazione. In base al suo valore il micro entrerà in un loop infinito della modalità scelta.
- **LCM**: procedura di *Low Computation Mode*, sez.5.2.2.1.
- **TCM**: procedura di *Test Computation Mode*, sez.5.2.2.3.
- **HCM**: procedura di *High Computation Mode*, sez.5.2.2.2.
- **Flag?**: test di condizione su una variabile di flag, modificata periodicamente da una routine tramite interrupt service.

### 5.2.1.2 Diagramma del modulo APP

Il comportamento del modulo APP relativo alla modalità **TCM** (sez.5.2.2.3) può essere rappresentato mediante il seguente diagramma di flusso:

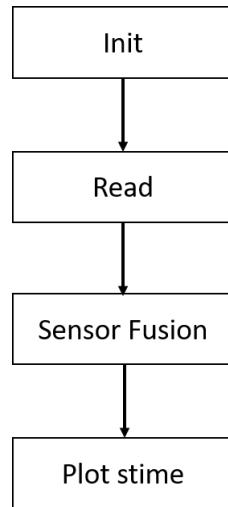


Figura 5.8: Rappresentazione del comportamento per il modulo Microcontrollore.

Dove:

- **Init**: rappresenta la fase di inizializzazione della procedura. Vengono inizializzate le costanti necessarie all'algoritmo di sensor fusion (sez.4.2), la variabile globale che definisce l'uso a 6DOF o 9DOF e infine i grafici da plottare successivamente.
- **Read**: rappresenta la fase di lettura dei dati trasferiti dal modulo microcontrollore e salvati in un file di log. Ogni riga del file, corrispondente ad un pacchetto TCM (sez.5.2.2.3), viene estratta e parsificata scomponendo la stringa in base ai separatori e i delimitatori utilizzati. Alla fine di questa fase si ottiene una matrice con i dati collezionati, per ogni sensore utilizzato, e una matrice con le stime dell'assetto elaborate dal microcontrollore.
- **Sensor Fusion**: rappresenta la fase di esecuzione dell'algoritmo di sensor fusion implementato (sez.4.2.3) a partire dalle matrici dei dati grezzi ottenuti nella fase precedente.
- **Plot stime**: vengono visualizzati gli angoli stimati dal microcontrollore e dal modulo APP al fine di completare l'analisi qualitativa dei due metodi (sez.6.3)

Per i dettagli implementativi si rimanda alla lettura dell'Appendice A.1.7.

## 5.2.2 Modalità di computazione

In questa sezione verranno illustrate le modalità computazionali implementate all'interno del sistema (sez.2.3) e i relativi pacchetti dati utilizzati.

### 5.2.2.1 Low Computation mode

Questa modalità ha lo scopo di alleggerire il carico computazionale del microcontrollore, da qui il nome di *low computation*. Infatti selezionandola l'algoritmo di *sensor fusion*, per la stima dell'assetto (sez.4.2), verrà eseguito dal modulo App, lasciando al microcontrollore la sola responsabilità di recuperare le misure effettuate dai sensori dell'IMU.

In questa modalità il pacchetto di dati da trasferire dal microcontrollore al modulo App ha la seguente struttura per la versione a 6DOF (giroscopio + accelerometro):

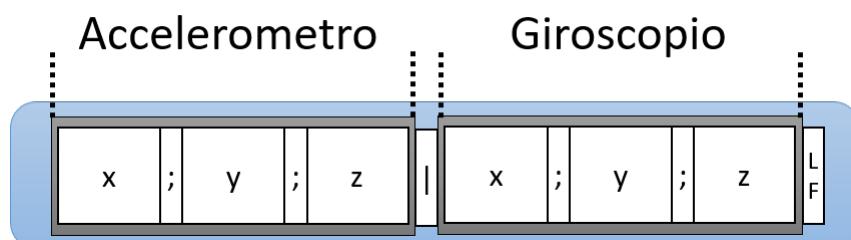


Figura 5.9: Rappresentazione del pacchetto dati utilizzato in *Low Computation Mode* a 6DOF

Con riferimento alla Fig.5.9, si mostra la tabella riepilogativa dei singoli campi:

Campo	Formattazione dati	Dimensione (bytes)	Descrizione
Accelerometro - X	x,xxx	5	Accelerazione misurata lungo l'asse X
Accelerometro - Y	y,yyy	5	” ” lungo l'asse Y
Accelerometro - Z	z,zzz	5	” ” lungo l'asse Z
		1	Delimitatore tra i dati dell'accelerometro e quelli del giroscopio
Giroscopio - X	xxx,xxx	7	Velocità angolare misurata attorno all'asse X
Giroscopio - Y	yyy,yyy	7	” ” misurata attorno all'asse Y
Giroscopio - Z	zzz,zzz	7	” ” misurata attorno all'asse Z
LF		1	Carattere ASCII per la codifica del comando “nuova riga”

Tabella 5.2: Tabella riepilogativa dei campi presenti nel pacchetto dati in LCM a 6DOF

A questi vanno aggiunti quattro campi ”;” utilizzati per separare le varie misure assiali dei sensori.

Sommando le dimensioni massime dei singoli campi mostrati in tab.5.2.2.1 si ottiene facilmente che la dimensione massima del pacchetto dati nella modalità *Low Computation* a 6DOF è di **42 bytes**.

Nell'uso a 9DOF viene aggiunto un campo per i dati provenienti dal magnetometro (sez.3.3), ottenendo la seguente struttura:

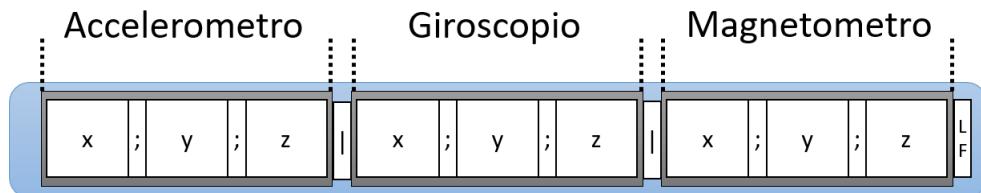


Figura 5.10: Rappresentazione del pacchetto dati utilizzato in *Low Computation Mode* a 9DOF

Con riferimento alla Fig.5.10, si mostra la tabella riepilogativa dei singoli campioni:

Campo	Formattazione dati	Dimensione (bytes)	Descrizione
Accelerometro - X	x,xxx	5	Accelerazione misurata lungo l'asse X
Accelerometro - Y	y,yyy	5	Accelerazione misurata lungo l'asse Y
Accelerometro - Z	z,zzz	5	Accelerazione misurata lungo l'asse Z
		1	Delimitatore tra i dati dei sensori
Giroscopio - X	xxx,xxx	7	Velocità angolare misurata attorno all'asse X
Giroscopio - Y	yyy,yyy	7	Velocità angolare misurata attorno all'asse Y
Giroscopio - Z	zzz,zzz	7	Velocità angolare misurata attorno all'asse Z
		1	Delimitatore tra i dati dei sensori
Magnetometro- X	x,xxx	5	Intensità del campo magnetico esterno sull'asse X
Magnetometro- Y	y,yyy	5	Intensità del campo magnetico esterno sull'asse Y
Magnetometro- Z	z,zzz	5	Intensità del campo magnetico esterno sull'asse Z
LF		1	Carattere ASCII per la codifica del comando "nuova riga"

Tabella 5.3: Tabella riepilogativa dei campi presenti nel pacchetto dati in LCM a 9DOF

Ancora una volta per determinare la dimensione totale del pacchetto, bisogna aggiungere i sei campi ";" che separano le varie misure assiali dei sensori. Così facendo si ottiene banalmente che la dimensione totale per la LCM in 9DOF è di **60**

**bytes.** Per i dettagli implementativi si rimanda alla lettura dell’appendice A.1.4.

### 5.2.2.2 High Computation mode

Questa modalità permette di assegnare al microcontrollore anche il compito di stimare l’assetto utilizzando la libreria *MotionFX* fornita da STM. Risulta evidente come questa modalità sia la duale della precedente (sez.5.2.2.1) in quanto aumenta il carico computazionale del microcontrollore, da cui il nome *High Computation*. Analogamente a quanto fatto per la modalità LCM, si mostra la struttura del pacchetto da trasferire dal microcontrollore al modulo APP per questa modalità:

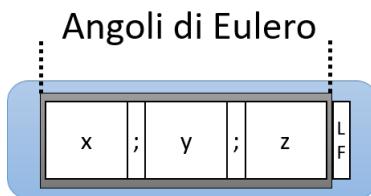


Figura 5.11: Rappresentazione del pacchetto dati utilizzato in *High Computation Mode*

Si noti che in questo caso non vi sono distinzioni tra le strutture dei pacchetti a 6DOF e 9DOF, infatti indipendentemente da quanti sensori verranno utilizzati il microcontrollore dovrà sempre stimare l’assetto e mandarlo sotto forma di *roll*, *pitch* e *yaw* (sez.4.1.2) al modulo APP. Con riferimento alla fig.5.11 si mostra la tabella riepilogativa dei campi:

Campo	Formattazione dati	Dimensione (bytes)	Descrizione
Yaw	xxx,x	5	Stima angolo di imbardata
Pitch	yyy,y	5	Stima angolo di beccheggio
Roll	zzz,z	5	Stima angolo di rollio
LF		1	Carattere ASCII per la codifica del comando "nuova riga"

Tabella 5.4: Tabella riepilogativa dei campi presenti nel pacchetto dati in HCM

Aggiungendo i due campi ";" utilizzati per separare gli angoli stimati, si ottiene che la dimensione del pacchetto dati da trasferire dal microcontrollore al modulo APP è di **18 bytes**. Si noti come in questa modalità, pur aumentando il carico computazionale richiesto dal microcontrollore, si abbattano le dimensioni del pacchetto da trasferire al numero minimo di bytes necessari per rappresentare le stime degli angoli d'assetto in modo significativo. Per i dettagli implementativi si rimanda alla lettura dell'appendice A.1.5.

### 5.2.2.3 Testing Computation mode

Quest'ultima modalità è stata pensata appositamente al fine di realizzare l'analisi qualitativa dell'assetto stimato (sez.6.3). La si può considerare come una modalità ibrida tra l'HCM e la LCM viste precedentemente, infatti in questo caso il pacchetto da trasferire dal microcontrollore al modulo APP include sia le misure dei sensori sia la stima dell'assetto.

In questa modalità il pacchetto di dati da trasferire dal microcontrollore al modulo App ha la seguente struttura per la versione a 6DOF:

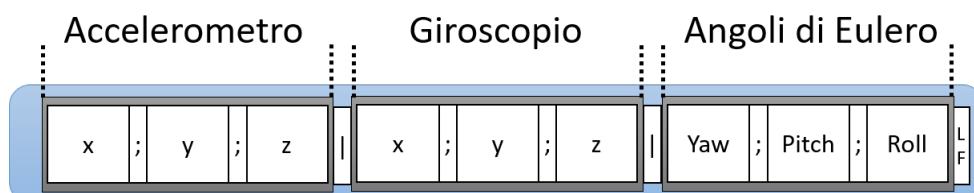


Figura 5.12: Rappresentazione del pacchetto dati utilizzato in *Test Computation Mode* a 6DOF

Con riferimento alla Fig.5.12, si mostra la tabella riepilogativa dei singoli campi:

Campo	Formattazione dati	Dimensione (bytes)	Descrizione
Accelerometro - X	x,xxx	5	Accelerazione misurata lungo l'asse X
Accelerometro - Y	y,yyy	5	Accelerazione misurata lungo l'asse Y
Accelerometro - Z	z,zzz	5	Accelerazione misurata lungo l'asse Z
		1	Delimitatore tra i dati dei sensori
Giroscopio - X	xxx,xxx	7	Velocità angolare misurata attorno all'asse X
Giroscopio - Y	yyy,yyy	7	Velocità angolare misurata attorno all'asse Y
Giroscopio - Z	zzz,zzz	7	Velocità angolare misurata attorno all'asse Z
		1	Delimitatore tra i dati dei sensori
Yaw	xxx,x	5	Stima angolo di imbardata
Pitch	yyy,y	5	Stima angolo di beccheggio
Roll	zzz,z	5	Stima angolo di rollio
LF		1	Carattere ASCII per la codifica del comando "nuova riga"

Tabella 5.5: Tabella riepilogativa dei campi presenti nel pacchetto dati in TCM a 6DOF

Ancora una volta aggiungendo sei campi ";" utilizzati per separare le misure assiali e gli angoli stimati si ottiene che la dimensione del pacchetto per questa modalità a 6DOF è **60 bytes**.

Nell'uso a 9DOF viene aggiunto un campo per i dati provenienti dal magnetometro (sez.3.3), ottenendo la seguente struttura:

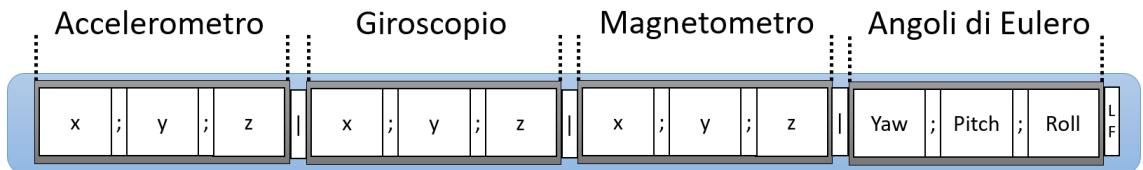


Figura 5.13: Rappresentazione del pacchetto dati utilizzato in *Test Computation Mode* a 9DOF

Con riferimento alla Fig.5.13, si mostra la tabella riepilogativa dei singoli campi:

Campo	Formattazione dati	Dimensione (bytes)	Descrizione
Accelerometro - X	x,xxx	5	Accelerazione misurata lungo l'asse X
Accelerometro - Y	y,yyy	5	Accelerazione misurata lungo l'asse Y
Accelerometro - Z	z,zzz	5	Accelerazione misurata lungo l'asse Z
		1	Delimitatore tra i dati dei sensori
Giroscopio - X	xxx,xxx	7	Velocità angolare misurata attorno all'asse X
Giroscopio - Y	yyy,yyy	7	Velocità angolare misurata attorno all'asse Y
Giroscopio - Z	zzz,zzz	7	Velocità angolare misurata attorno all'asse Z
		1	Delimitatore tra i dati dei sensori
Magnetometro- X	x,xxx	5	Intensità del campo magnetico esterno sull'asse X
Magnetometro- Y	y,yyy	5	Intensità del campo magnetico esterno sull'asse Y
Magnetometro- Z	z,zzz	5	Intensità del campo magnetico esterno sull'asse Z
		1	Delimitatore tra i dati dei sensori e l'assetto stimato
Yaw	xxx,x	5	Stima angolo di imbardata
Pitch	yyy,y	5	Stima angolo di beccheggio
Roll	zzz,z	5	Stima angolo di rollio
LF		1	Carattere ASCII per la codifica del comando "nuova riga"

Tabella 5.6: Tabella riepilogativa dei campi presenti nel pacchetto dati in TCM a 9DOF

Infine aggiungendo otto campi ";" utilizzati per separare le misure assiali e gli angoli stimati si ottiene che la dimensione del pacchetto per questa modalità a 9DOF è **78 bytes**.

Quindi la TCM a 9DOF risulta essere la modalità con il maggiore carico computazionale richiesto al microcontrollore e con la dimensione maggiore dei pacchetti da trasferire dal micro al modulo APP. Per i dettagli implementativi si rimanda alla lettura dell'appendice A.1.6.

# Capitolo 6

## Analisi e validazione dei risultati

### 6.1 Analisi temporale

Questo tipo di analisi è stata effettuata al fine di determinare la frequenza massima entro la quale il microcontrollore è in grado di eseguire tutte le operazioni basilari, rappresentate in Fig.6.1. Ovvero per rispondere alla seguente domanda:

*Qual'è la finestra temporale minima sufficiente a garantire che il microcontrollore sia in grado di eseguire correttamente le funzioni basilari?*

Dove per funzioni basilari si intendono le operazioni di:

- **Lettura** dei dati grezzi dall'IMU tramite, indicato con  $T_{RX\_I2C}$
- **Elaborazione** dei dati grezzi (solo per le modalità *HCM* e *TCM*), indicato con  $T_{MotionFX}$
- **Trasmissione** dei dati verso il modulo *App* (si veda 2.3), indicato con  $T_{TX\_USB}$

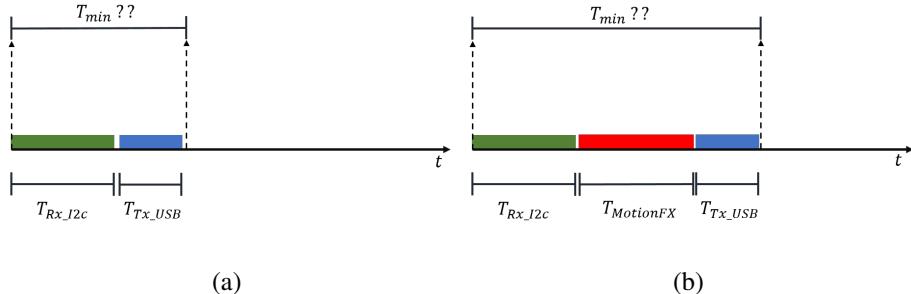


Figura 6.1: In 6.1(a) le operazioni basilari per la modalità **LCM**, in 6.1(b) le operazioni basilari per le modalità **HCM** e **TCM**

Per far ciò è opportuno analizzare singolarmente i due tempi, considerando le relative informazioni e la velocità del mezzo di comunicazione.

### 6.1.1 Analisi del tempo di lettura dei dati grezzi dall'IMU

Le informazioni riguardanti le grandezze fisiche misurate dai sensori integrati nell'IMU (Cap.3), tramite I2C (Cap.5.1.2.1), su un singolo asse del *b-frame* (Cap.3.4), sono rappresentate attraverso **16 bit**.

Di conseguenza il tempo di lettura  $T_{RX\_I2C}$  è dato dalla seguente equazione:

$$T_{RX\_I2C} = N_{sensori} \cdot N_{assi} \cdot T_{i\_asse} \quad (6.1)$$

Dove:

- $N_{sensori}$  sono il numero di sensori utilizzati
  - $N_{assi}$  sono il numero di assi utilizzati dai sensori
  - $T_{i\_asse}$  è il tempo di lettura del singolo asse

Dunque il problema si riduce alla determinazione di  $T_{i\_asse}$ . Per come è implementata l'IMU, la lettura di un singolo asse si completa leggendo due registri da **8 bit**.

$$T_{i, \text{asse}} = 2 \cdot T_{BX, 8bit} \quad (6.2)$$

Per stimare  $T_{RX\_8bit}$  si è programmato il microcontrollore in modo tale da leggere 1000 volte un registro da 8 bit e misurare il tempo trascorso leggendo i *tick* di sistema. Per il codice si rimanda alla lettura dell'App.A.2.1.

Eseguendo il test svariate volte, si è osservato che il tempo necessario per leggere 1000 volte un registro da 8 bit è pari a  $101ms$ . Questo implica un *goodput* di  $80Kb/s$  pari al 20% della capacità del canale di comunicazione, che risulta essere  $400Kb/s$  essendo un I2C in *fast-mode*. Questo calo è dovuto a numerosi fattori tra i quali l'overhead del protocollo e la velocità del microcontrollore.

Quindi andando a sostituire il valore stimato per la lettura di 8 bit nell'Eq.6.1 si ottiene, per l'uso dell'IMU a 9DOF:

$$T_{RX\_I2C} = 3 \cdot 3 \cdot (2 \cdot T_{RX\_8bit}) = 3 \cdot 3 \cdot (2 \cdot 101\mu s) = 1,8ms \quad (6.3)$$

Mentre per l'uso a 6DOF:

$$T_{RX\_I2C} = 3 \cdot 2 \cdot (2 \cdot T_{RX\_8bit}) = 3 \cdot 2 \cdot (2 \cdot 101\mu s) = 1,2ms \quad (6.4)$$

### 6.1.2 Analisi del tempo di trasmissione dei dati tramite USB

Il tempo necessario per trasmettere un pacchetto, dal modulo *microcontrollore* al modulo *App* (Cap.2.2) mediante il canale USB (Cap.5.1.2.2), è dato dalla seguente equazione:

$$T_{TX\_USB} = \frac{dim(pack)}{goodput} \quad (6.5)$$

Per stimare il *goodput* si è programmato il microcontrollore in modo da inviare 1000 volte un pacchetto contenente un *id* (univoco e incrementale) e un contatore incrementato quando il buffer di trasmissione risulta essere ancora occupato. Per il codice si rimanda alla lettura dell'App.A.2.2. Eseguendo il test numerose volte, si è osservato che il tempo necessario per trasmettere 1000 pacchetti, per un totale di 8574 Bytes, è di circa  $57ms$ . Questo implica un *goodput* di  $1,2Mb/s$  pari al 10% della capacità del canale di comunicazione che, essendo un USB 1.1, risulta essere  $12Mb/s$ . Ancora una volta questo calo di prestazioni è dovuto a numerosi fattori tra i quali l'overhead del protocollo, l'implementazione della libreria STM utilizzata e la velocità del microcontrollore.

Più complicato è il discorso riguardante la dimensione dei pacchetti da trasmettere, questi infatti sono strettamente correlati a:

- **La modalità di computazione** selezionata (Cap.5.2.2)
- **Il numero di sensori** utilizzati
- **La codifica** dell'informazione

Con riferimento alle possibili combinazioni dei fattori precedenti e alle rispettive strutture dei pacchetti, dettagliate nel Cap.5.2.2, si hanno le seguenti dimensioni:

**1. LCM:**

- (a)  $6DOF = 48$  Bytes
- (b)  $9DOF = 63$  Bytes

**2. HCM = 21 Bytes**

**3. TCM:**

- (a)  $6DOF = 63$  Bytes
- (b)  $9DOF = 84$  Bytes

Sostituendo questi valori e la stima del *goodput* del canale all'Eq.6.5, si ottengono le stime dei diversi tempi di trasmissione:

**1. LCM:**

- (a)  $6DOF: T_{TX\_USB} = \frac{48Bytes}{1,2Mbit/s} = 0,32ms$
- (b)  $9DOF : T_{TX\_USB} = \frac{63Bytes}{1,2Mbit/s} = 0,42ms$

**2. HCM :  $T_{TX\_USB} = \frac{21Bytes}{1,2Mbit/s} = 0,16ms$**

**3. TCM:**

- (a)  $6DOF : T_{TX\_USB} = \frac{63Bytes}{1,2Mbit/s} = 0,42ms$
- (b)  $9DOF : T_{TX\_USB} = \frac{84Bytes}{1,2Mbit/s} = 0,56ms$

### 6.1.3 Conclusioni analisi temporale

Sommando i tempi stimati nei paragrafi precedenti (Cap.6.1.1 e Cap.6.1.2), si ottengono le diverse finestre temporali minime affinché, il microcontrollore riesca a completare le funzioni basilari:

**1. LCM:**

- (a)  $6DOF: T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,2ms + 0,32ms = 1,52ms$
- (b)  $9DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,8ms + 0,42ms = 2,22ms$

## 2. HCM :

- (a)  $6DOF: T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,2ms + 0,16ms = 1,36ms$
- (b)  $9DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,8ms + 0,16ms = 1,96ms$

## 3. TCM:

- (a)  $6DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,2ms + 0,42ms = 1,62ms$
- (b)  $9DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,8ms + 0,56ms = 2,36ms$

Da notare che tali stime non tengono conto di alcune latenze legate all'esecuzione del codice all'interno del microprocessore, per questo motivo si ritiene opportuno sovrastimare le finestre temporali e sottostimare le frequenze massime, in modo da avere margini di sicurezza e stabilità maggiori.

Per le modalità *HCM* e *TCM* si deve tener conto anche del tempo di computazione della libreria *Motion FX* per la stima della dell'assetto a partire dai dati grezzi che, nel caso peggiore è dichiarato essere  $3,2ms$  [25] per il microcontrollore utilizzato nell'ambito di questa tesi.

Quindi le stime aggiornate risultano essere:

## 1. LCM:

- (a)  $6DOF: T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,2ms + 0,32ms = 1,52ms$
- (b)  $9DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} = 1,8ms + 0,42ms = 2,22ms$

## 2. HCM :

- (a)  $6DOF: T_{min} = T_{RX\_I2C} + T_{TX\_USB} + T_{MotionFX} = 1,2ms + 0,16ms + 3,2ms = 4,56ms$
- (b)  $9DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} + T_{MotionFX} = 1,8ms + 0,16ms + 3,2ms = 5,16ms$

## 3. TCM:

- (a)  $6DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} + T_{MotionFX} = 1,2ms + 0,42ms + 3,2ms = 4,82ms$
- (b)  $9DOF : T_{min} = T_{RX\_I2C} + T_{TX\_USB} + T_{MotionFX} = 1,8ms + 0,56ms + 3,2ms = 5,56ms$

Che corrispondono alle seguenti frequenze di lavoro massime:

**1. LCM:**

- (a)  $6DOF: f = 657Hz$
- (b)  $9DOF : f = 450Hz$

**2. HCM :**

- (a)  $6DOF: f = 219Hz$
- (b)  $9DOF : f = 193Hz$

**3. TCM:**

- (a)  $6DOF : f = 207Hz$
- (b)  $9DOF : f = 179Hz$

## 6.2 Analisi dello zero-offset

Quest’analisi è stata effettuata al fine di stimare lo **zero-offset** dell’accelerometro e del giroscopio. Tale fenomeno consiste nella deviazione dei dati in output dai sensori (5.1.2.1) rispetto al valore ideale atteso quando il sensore non subisce forze esterne. L’origine di questo errore è strettamente legata agli stress meccanici subiti dall’IMU nei processi di realizzazione.

Relativamente ai sopra citati sensori, si hanno le seguenti definizioni di zero-offset:

- **zero-g offset:** nel caso ideale un accelerometro posto in posizione orizzontale e immobile misura  $0\ g$  sugli assi X e Y mentre  $1\ g$  sull’asse Z. Lo *zero-g* è la deviazione del valore letto in uscita rispetto al valore ideale.
- **zero-rate offset:** nel caso ideale un giroscopio immobile misura  $0\ dps$  su tutti gli assi. Lo *zero-g* è la deviazione del valore letto in uscita rispetto al valore ideale.

### 6.2.1 Stima dello zero-g offset

Per stimare lo zero-g offset si è posto immobile l’IMU e si sono raccolti i dati letti dall’accelerometro per un totale di due minuti. Tale procedura è stata ripetuta

tre volte ponendo ogni volta l'IMU parallelo rispetto ad un'asse (Cap.3.4), come mostrato in Fig.6.2.1:

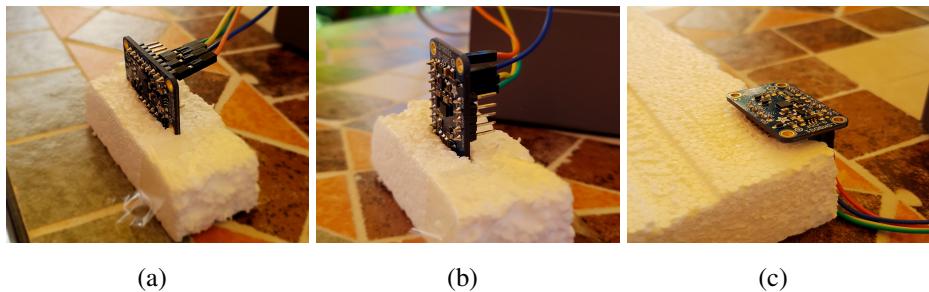


Figura 6.2: in 6.2(a) l'IMU posto parallelamente all'asse  $X$ , in 6.2(b) l'IMU posto parallelamente all'asse  $Y$  e infine in 6.2(c) l'IMU posto parallelamente all'asse  $Z$

Quindi per ogni lettura si è determinata la norma euclidea, dalla quale banalmente si ottiene il modulo dello zero-g offset sottraendo 1. Tale valore è da considerarsi come modulo e non sui singoli assi in quanto il dispositivo potrebbe non essere stato posizionato perfettamente parallelo all'asse in questione. In Fig.6.2.1 vengono mostrati i valori degli zero-offset per i campioni catturati nei primi 7 secondi per ognuno dei tre assi:

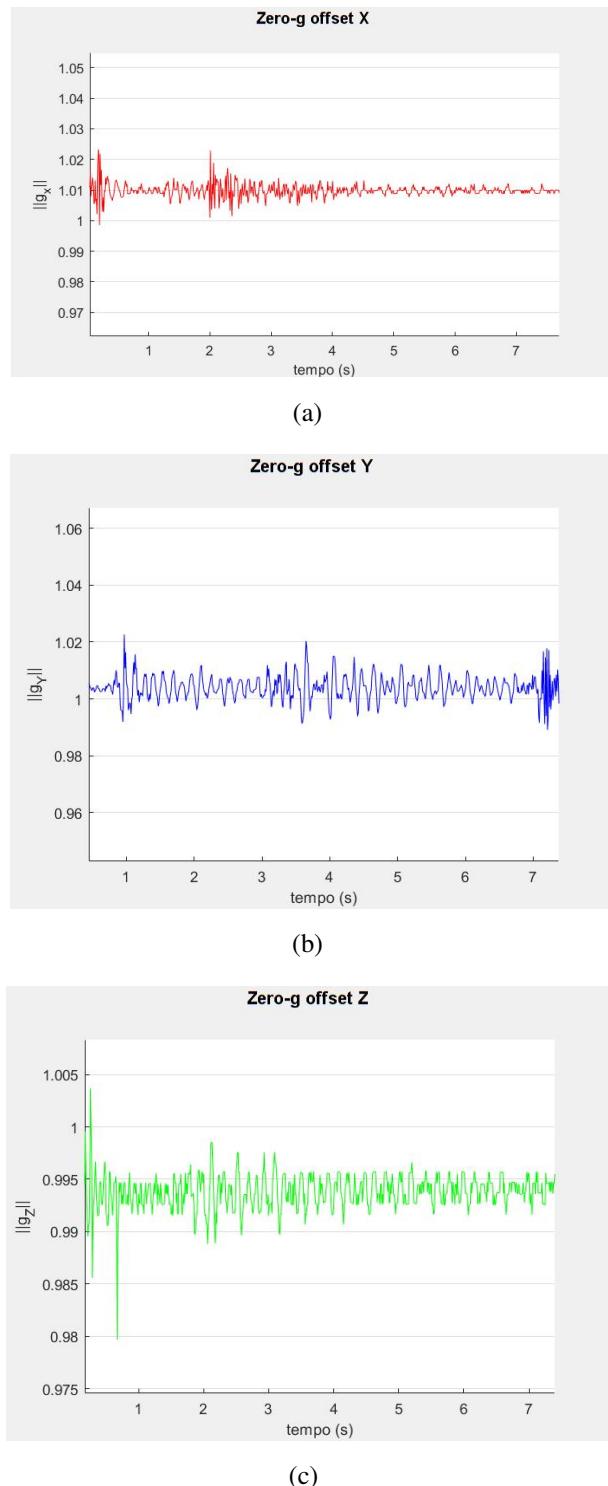


Figura 6.3: valori degli zero-g offset calcolati rispettivamente sull'asse X 6.3(a), sull'asse Y 6.3(b) e sull'asse 6.3(c)

Nella Tab.6.2.1 si riportano i valori di *Media*, *Max* e le percentuali di occorrenze dei valori compresi nell’intervallo (0,1) e maggiori di 1.

Asse	Media (G)	Max (G)	val>1	0;val>1
x	1,0096	1,0233	99	1
y	1,0036	1,0266	99	1
z	0,9936	1,0037	0,02	99,08

Tabella 6.1: Media, Max e percentuali di occorrenze dei valori compresi nell’intervallo (0,1) e maggiori di 1 dei dati mostrati in Fig.6.3(a)

### 6.2.2 Stima dello zero-rate offset

Per stimare lo zero-rate offset si è posto immobile l’IMU (come in Fig.6.2.1) e si sono raccolti i dati letti dal giroscopio per un totale di due minuti. I valori ottenuti, per ognuno dei tre assi, sono mostrati in Fig.6.4:

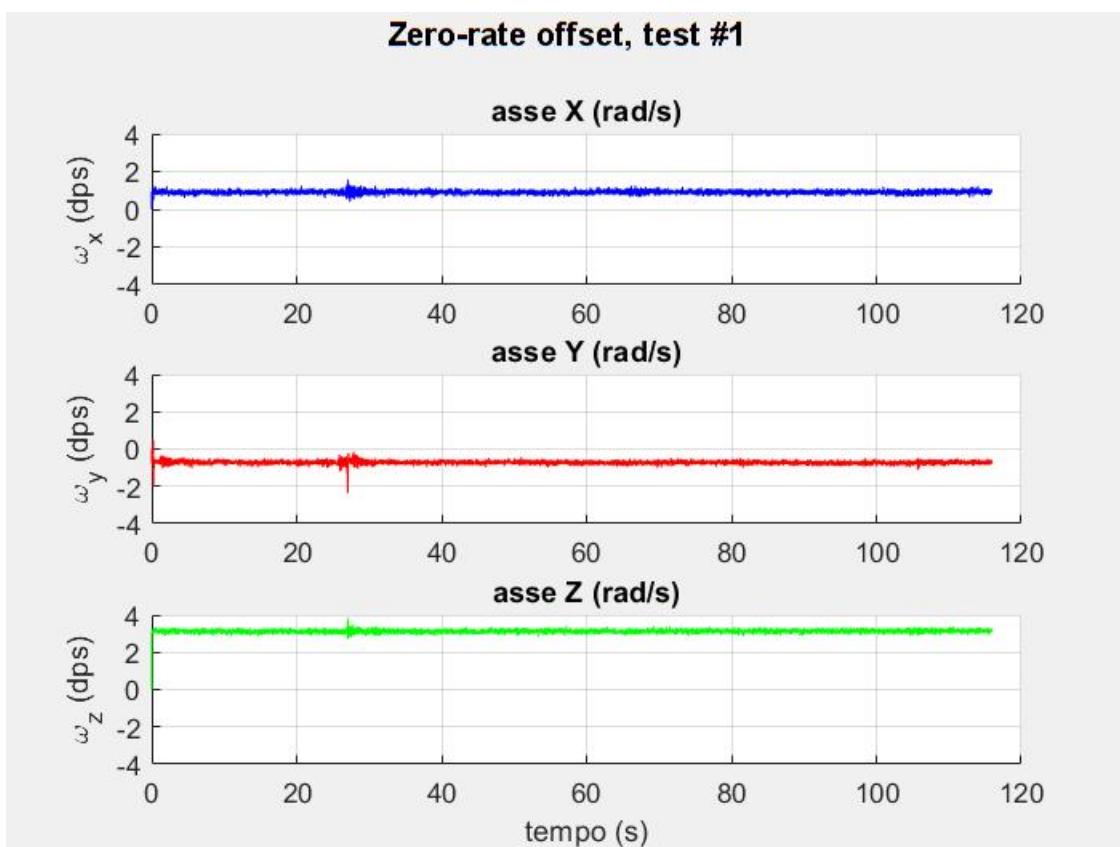


Figura 6.4: valori degli zero-rate offset per i campioni catturati in 2 minuti rispettivamente sull’asse X, Y e Z.

Dai valori mostrati in Fig.6.4 si evince facilmente la presenza di un offset significativo sull’asse Z da cui ne consegue che, pur rimanendo immobile il giroscopio ha *”la sensazione”* di essere ruotato intorno all’asse in questione con una velocità angolare significativa.

Nella Tab.6.2.2 si riportano invece i valori di *media* e *varianza* per i campioni mostrati in Fig.6.4:

Asse	Media (dps)	Varianza
x	0,9055	0,0079
y	-0,7203	0,0091
z	3,1291	0,0064

Tabella 6.2: Media e varianza per i valori di zero-rate offset mostrati in Fig.6.4

### 6.2.3 Conclusioni stima dello zero-offset

L’analisi appena effettuata ha mostrato come i dati grezzi provenienti dall’unità di misura inerziale (Cap.3) siano affetti dallo zero-offset.

L’accelerometro non risulta subire molto questo fenomeno, infatti i valori medi dei moduli del vettore gravitazionale si discostano al più dello 0,96% (6.2.1) sull’asse delle  $X$  e il valore massimo si ha in corrispondenza dell’asse  $Y$  che supera il valore ideale di  $0,0266G$ .

Ben diversa è la situazione dello zero-rate offset che risulta essere molto significativo soprattutto sull’asse  $Z$  dove si ha una media di  $3,1291dps$ . I valori di varianza bassi sottolineano come tale errore sia relativamente costante nel tempo fornendo quindi un punto di partenza per una sua modellizzazione.

A valle di queste considerazioni si vuole evidenziare come l’uso dei dati grezzi provenienti dall’IMU per un’elaborazione che permetta la stima dell’assetto, debba essere preceduto da un’attività di compensazione di tali offset. Compensazione che può, in alcuni casi, trasformarsi in un vero e proprio lavoro di caratterizzazione e modellizzazione dell’errore all’interno dell’algoritmo di elaborazione.

## 6.3 Analisi qualitativa dell’assetto stimato

Quest’analisi ha lo scopo di verificare la correttezza dell’algoritmo di *sensor fusion* implementato (sez.4.2) e della libreria *MotionFX* [25]. In particolare si vuole confrontare la stima dello *yaw* (sez.4.1.2) a seguito di varie sollecitazioni. Questo perché la stima dell’imbardata è, attualmente, quella più utile per identificare gli angoli assunti dall’operatore nel tragitto tra un nodo e l’altro (cap.2).

Per fare questo si è realizzato un ”sistema ausiliario” ad hoc in grado di ruotare l’unità di misura inerziale ripetutamente e di un certo angolo. Tale sistema è composto da:

- Un Arduino duemilanove
- Un servo motore MG995

Ed è rappresentato dalla seguente figura:

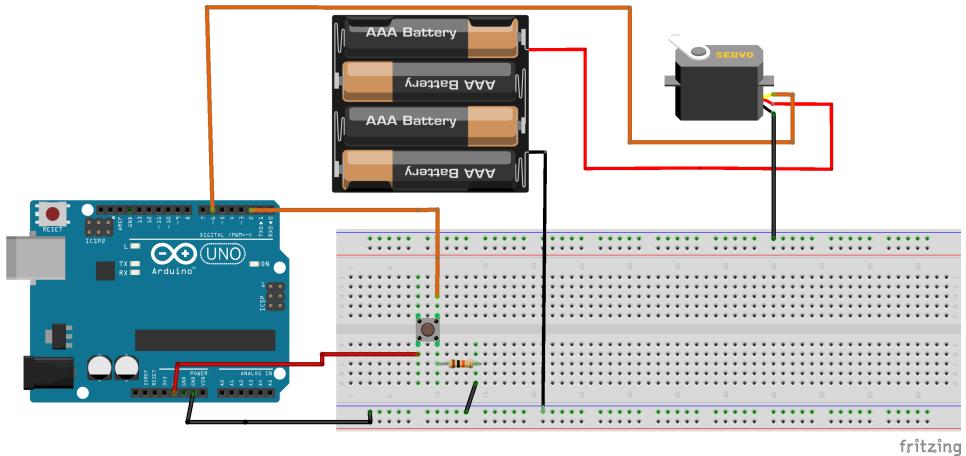


Figura 6.5: Schema hardware del sistema ausiliario realizzato per eseguire l’analisi qualitativa

Alla pressione del bottone mostrato in Fig.6.5, il codice all’interno di Arduino (App.A.2.3) manda un impulso al servo motore grazie al quale il servo ruota di  $90^\circ$ . Quindi Arduino attende 500 ms prima di inviare un’ulteriore segnale affinché il servo torni alla posizione iniziale. Questa procedura viene ripetuta dieci volte aspettando 1 secondo tra le prime cinque rotazioni di  $\pm 90^\circ$  e le ultime. Il modulo App elabora i dati grezzi letti dal modulo Microcontrollore (sez.5.1) a seguito di questi movimenti e applica l’algoritmo di *Sensor fusion* implementato.

Le stime dello *yaw* ottenute dai due moduli, a partire dagli stessi dati grezzi, sono mostrate dalle seguente figura:

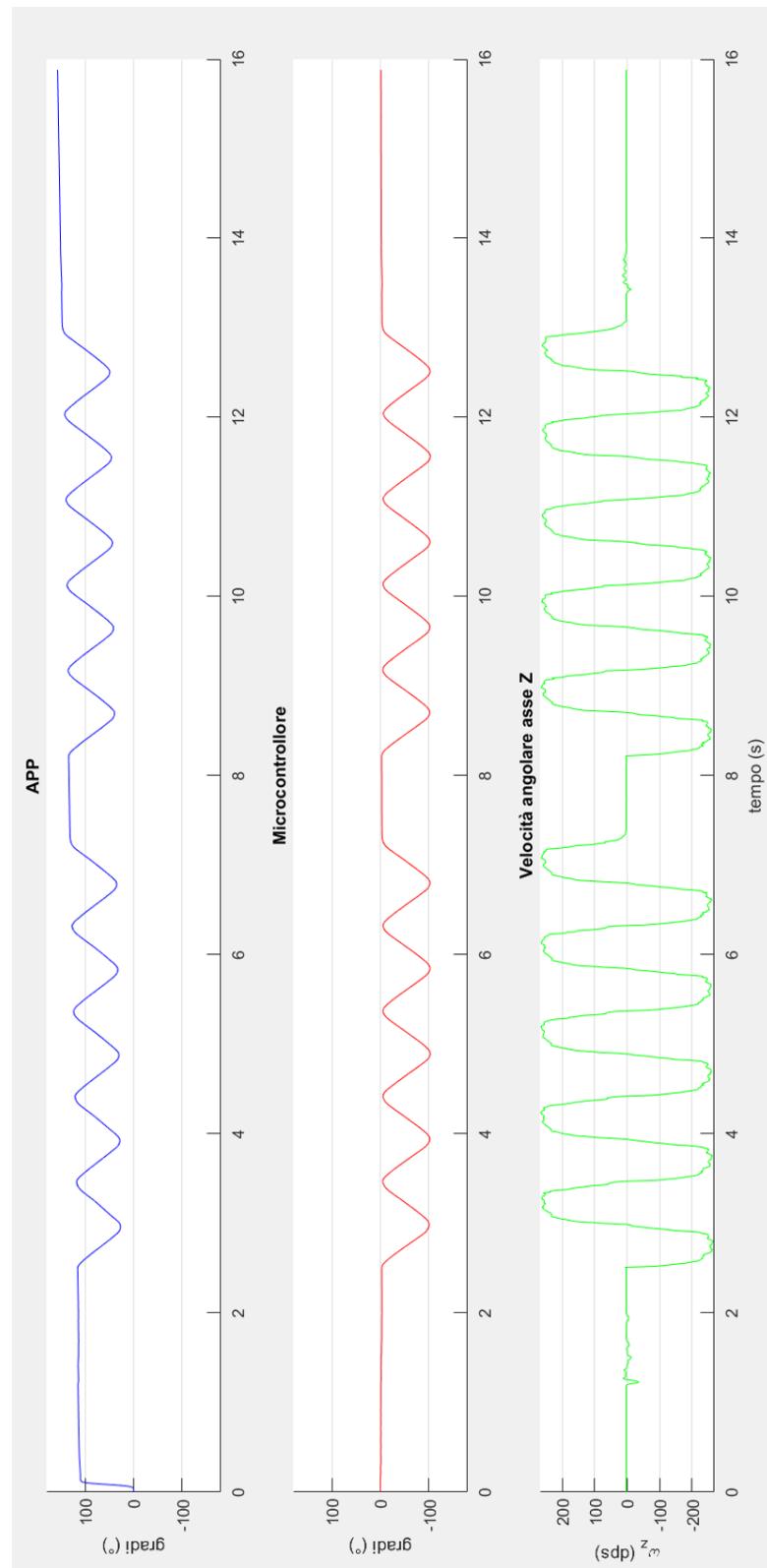


Figura 6.6: Risultati analisi qualitativa

Come si può notare dalla Fig.6.6, la risposta generale dei due algoritmi è buona. Infatti entrambi rispondono correttamente ai movimenti rotatori effettuati intorno all’asse  $Z$  dell’IMU facendo oscillare lo  $yaw$  di  $\pm 90^\circ$  rispetto all’angolo di partenza (da notare che per il modulo APP è circa  $110^\circ$  mentre per il micro è  $0^\circ$ ).

Tuttavia si nota facilmente come la stima da parte del modulo APP diverga sempre più, muovendo verso l’alto il range dei valori picco-picco. Tale variazione è costante e rappresentata dalla seguente figura:

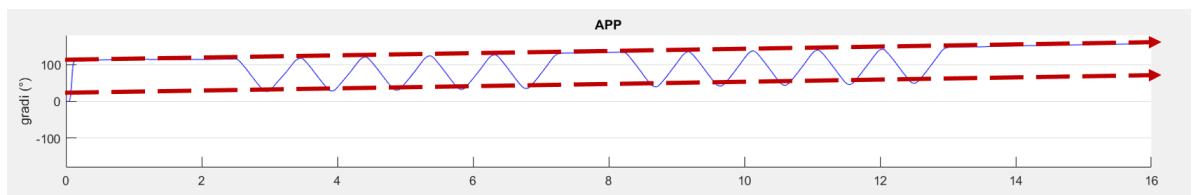


Figura 6.7: Retta di divergenza della stima dello  $yaw$  eseguita dal modulo APP.

A partire da un angolo stimato di  $110^\circ$  lo  $yaw$  dopo dieci rotazioni di  $\pm 90^\circ$  effettuate in  $\approx 15\text{sec}$  lo  $yaw$  finale risulta essere  $157^\circ$ . Da tali valori si calcola facilmente che il coefficiente angolare delle rette di divergenza mostrate in Fig.6.7 è di  $3,13$ . Questo risultato è coerente con quanto emerso nell’analisi dello zero-rate (sez.6.2.2), ovvero che lungo l’asse  $z$  si ha un *drift* di valore medio pari a  $\approx 3,12\text{dps}$ .

Alla luce di questa analisi è facile capire come l’algoritmo di *Sensor fusion* (sez.4.2) implementato e utilizzato dal modulo APP non tenga conto di questo fenomeno, cosa che avviene invece all’interno della libreria *MotionFX* utilizzata dal modulo Microcontrollore poiché i valori picco-picco restano confinati all’interno dell’intervallo  $[0^\circ, -90^\circ]$  come mostrato dalla Fig.6.6.

## **Capitolo 7**

### **Conclusioni e prospettive future**

# Appendice A

## Script

### A.1 Operativi

#### A.1.1 Script per la lettura dei valori misurati da Accelerometro e Giroscopio tramite I2C

Di seguito si riporta il codice utilizzato per leggere i dati misurati dall'accelerometro e dal giroscopio utilizzando un canale I2C in fast-mode.

---

```
LSM9DS1_XLG_READ LSM9DS1_Read_XLG(MFX_input_t *data_in,
    int samples)
{
    int16_t pData[6]={0,0,0,0,0,0};
    int16_t data=0;
    int8_t regadd;
    int i,j,k;
    float divisor=1000;
    float sensitivityXL= (float) LSM9DS1_get_Sens_XL
        ();
    float sensitivityG= (float) LSM9DS1_get_Sens_G();

    uint8_t hi;
    uint8_t lo;
```

```
/* Read output registers from LSM9DS1_OUT_X_L_G
   to LSM9DS1_OUT_Z_XL. */
for (i = 0; i < samples; i++)
{
    regadd=LSM9DS1_OUT_X_L_G;

    //read Gyroscope output
    for (j = 0; j < 3; j++)
    {
        if( !I2C_ReadData(LSM9DS1_I2C_BADD,
                            regadd, &lo,1))
        {
            return 0;
        }
        regadd++;
        if( !I2C_ReadData(LSM9DS1_I2C_BADD,
                            regadd, &hi,1))
        {
            return 0 ;
        }
        regadd++;

        data = ((uint16_t)hi << 8) | (
            uint16_t)lo;
        pData[j]=data;

    }
    /* Format the data. */
    data_in->gyro[0] = (pData[0] * sensitivityG
        )/divisor;
    data_in->gyro[1] = (pData[1] * sensitivityG
        )/divisor;
    data_in->gyro[2] = ((pData[2] *
        sensitivityG )/divisor);

    regadd=LSM9DS1_OUT_X_L_XL;
```

```
//read Accelerometer output
for (j = 3; j < 6; j++)
{
    if( !I2C_ReadData(LSM9DS1_I2C_BADD,
                         regadd, &lo,1))
    {
        return 0;
    }
    regadd++;
    if( !I2C_ReadData(LSM9DS1_I2C_BADD,
                         regadd, &hi,1))
    {
        return 0 ;
    }
    regadd++;

    data = ((uint16_t)hi << 8) | (
        uint16_t)lo;
    pData[j]=data;

}
/* Format the data. */
data_in->acc[0] = (pData[3] * sensitivityXL
                     )/divisor;
data_in->acc[1] = (pData[4] * sensitivityXL
                     )/divisor;
data_in->acc[2] = (pData[5] * sensitivityXL
                     )/divisor;
}
return 1;
}
```

### A.1.2 Script di inizializzazione del micro

Di seguito si riporta il codice utilizzato per stimare il tempo di lettura di dati provenienti dall’unità di misura inerziale utilizzando un canale I2C in fast-mode.

---

```
//Test I2C speed
while(1) {
    if(id<1000) {

        if(!I2C_ReadData(LSM9DS1_I2C_BADD, LSM9DS1_WHO_AM_I, &
                           result,1)) {
            if( result == LSM9DS1_WHO_AM_I_VALUE)
                id++;
            }
        else
        {
            lost++;
        }

        if(id==1000) {
            t1=HAL_GetTick()-t0;
            HAL_Delay(2);
            snprintf(data, 10, "%Lu",t1);
            CDC_Transmit_FS((uint8_t *)data,strlen(data));
        }
    }
}
```

### A.1.3 Script per la stima del tempo di trasmissione di dati al modulo App

Di seguito si riporta il codice utilizzato per inizializzare il micro.

---

```
.
.
.

#define ComputationMode      TEST_COMPUTATION_MODE
#define DOF_MODE             MODE_6DOF
.
.
```

```
.  
int main(void)  
{  
    .  
    .  
    .  
    LSM9DS1_State_Connection testConnection=  
        LSM9DS1_ERROR;  
    testConnection=LSM9DS1_IsConnected();  
    LSM9DS1_XLG_TurnOff();  
  
    if (DOF_MODE==MODE_9DOF) {  
        LSM9DS1_XLG_READ start= LSM9DS1_XLGM_Start  
            (119, 40, 245, 8, 8);  
        MotionFX_manager_init(MODE_9DOF);  
    }  
    else{  
        LSM9DS1_XLG_READ start= LSM9DS1_XLG_Start  
            (119, 245, 2);  
        MotionFX_manager_init(MODE_6DOF);  
    }  
    .  
    .  
    .  
}
```

#### A.1.4 Script Low Computation Mode

Di seguito si riporta il codice utilizzato per implementare la funzione di Low Computation Mode all'interno del modulo Microcontrollore.

---

```
void LowComputationMode() {  
  
    HAL_TIM_Base_Start_IT(&htim3);  
    while(1)
```

```
{  
    if (flag==1)  
    {  
  
        HAL_NVIC_DisableIRQ(TIM3_IRQn);  
  
        if (DOF_MODE == MODE_6DOF)  
        {  
  
            LSM9DS1_Read_XLG(&data_in,1);  
  
            snprintf(data, 64, "%.3f;%.3f  
                    ;%.3f;%3.1f;%3.1f;%3.1f\n",  
                    data_in.acc[0], data_in.acc[1],  
                    data_in.acc[2],  
                    data_in.gyro[0],data_in.gyro  
                    [1],data_in.gyro[2]);  
        }  
        else  
        {  
            LSM9DS1_Read_XLGM(&data_in,1);  
            snprintf(data, 64, "%.3f;%.3f  
                    ;%.3f;%3.1f;%3.1f;%3.1f;%3.1  
                    f;%3.1f;%3.1f\n",  
                    data_in.acc[0], data_in.acc[1],  
                    data_in.acc[2],  
                    data_in.gyro[0],data_in.gyro  
                    [1],data_in.gyro[2],  
                    data_in.mag[0],data_in.mag[1],  
                    data_in.mag[2]);  
        }  
  
        //transmit data  
        CDC_Transmit_FS((uint8_t *)data,  
                         strlen(data));  
        flag=0;  
    }
```

```
    HAL_NVIC_EnableIRQ(TIM3_IRQn);  
}  
}  
}
```

### A.1.5 Script High Computation Mode

Di seguito si riporta il codice utilizzato per implementare la funzione di High Computation Mode all'interno del modulo Microcontrollore.

---

```
void HighComputationMode () {  
  
    HAL_TIM_Base_Start_IT(&htim3);  
    HAL_TIM_Base_Start_IT(&htim2);  
    while (1)  
    {  
        if (flag==1)  
        {  
  
            HAL_NVIC_DisableIRQ(TIM3_IRQn);  
  
            //prepare packet  
            if (DOF_MODE == MODE_6DOF)  
            {  
                LSM9DS1_Read_XLG (&data_in,1);  
                MotionFX_manager_run (&data_in,&  
                    data_out,MFX_DELTATIME);  
                sprintf(data, 25, "%3.1f;%3.1f  
                    ;%3.1f\n",  
                    data_out.rotation_6X[0],  
                    data_out.rotation_6X[1],  
                    data_out.rotation_6X[2]);  
            }  
            else  
            {  
                //  
            }  
        }  
    }  
}
```

```
LSM9DS1_Read_XLGM(&data_in, 1);
if(flag2==1) {
    HAL_NVIC_DisableIRQ(TIM2_IRQn);
    mag_in.mag[0]=data_in.mag
        [0]/50;
    mag_in.mag[1]=data_in.mag
        [1]/50;
    mag_in.mag[2]=data_in.mag
        [2]/50;
    mag_in.time_stamp=0;
    MotionFX_MagCal_run(&mag_in);
    MotionFX_MagCal_getParams(&
        mag_out);
    if(mag_out.cal_quality != MFX_MAGCALPOOR) {
        data_in.mag[0]=mag_in.mag[0]-
            mag_out.hi_bias[0];
        data_in.mag[1]=mag_in.mag[1]-
            mag_out.hi_bias[1];
        data_in.mag[2]=mag_in.mag[2]-
            mag_out.hi_bias[2];
    }
    flag2=0;
    HAL_NVIC_EnableIRQ(TIM2_IRQn);
}
MotionFX_manager_run(&data_in, &
    data_out,MFX_DELTATIME);
snprintf(data, 25, "%3.1f;%3.1f
    ;%3.1f\n",
    data_out.rotation_9X[0],
    data_out.rotation_9X[1],
    data_out.rotation_9X[2]);
}

//transmit data
```

```
        CDC_Transmit_FS( (uint8_t *)data,
                           strlen(data));
        flag=0;
        HAL_NVIC_EnableIRQ(TIM3_IRQn);
    }

}
```

### A.1.6 Script Test Computation Mode per il modulo Microcontrollore

Di seguito si riporta il codice utilizzato per implementare la funzione di Test Computation Mode all'interno del modulo Microcontrollore.

---

```
void TestComputationMode () {

    HAL_TIM_Base_Start_IT(&htim3);
    HAL_TIM_Base_Start_IT(&htim2);

    while(1)
    {
        if(flag==1)
        {

            HAL_NVIC_DisableIRQ(TIM3_IRQn);

            //prepare packet
            if(DOF_MODE == MODE_6DOF)
            {
                LSM9DS1_Read_XLG(&data_in,1);
                MotionFX_manager_run (&data_in,&
                                      data_out,MFX_DELTATIME);
            }
        }
    }
}
```

```
snprintf(data, 128, "%.3f;%.3f
;.3f|.3f;.3f;%.3f|%.1f
;%.1f;%.1f\n",
data_in.acc[0], data_in.acc[1],
data_in.acc[2],
data_in.gyro[0], data_in.gyro
[1], data_in.gyro[2],
data_out.rotation_6X[0],
data_out.rotation_6X[1],
data_out.rotation_6X[2]);
}

else
{
    LSM9DS1_Read_XLGM(&data_in, 1);
    if(flag2==1)
    {
        HAL_NVIC_DisableIRQ(
            TIM2_IRQn);
        mag_in.mag[0]=data_in.mag
            [0]/50;
        mag_in.mag[1]=data_in.mag
            [1]/50;
        mag_in.mag[2]=data_in.mag
            [2]/50;
        mag_in.time_stamp=0;
        MotionFX_MagCal_run(&
            mag_in);
        MotionFX_MagCal_getParams
            (&mag_out);
        if(mag_out.cal_quality !=
            MFX_MAGCALPOOR) {
            data_in.mag[0]=mag_in.mag
                [0]-mag_out.hi_bias
                [0];
            data_in.mag[1]=mag_in.mag
                [1]-mag_out.hi_bias
                [1];
        }
    }
}
```

```
[1];
data_in.mag[2]=mag_in.mag
[2]-mag_out.hi_bias
[2];
}

flag2=0;
HAL_NVIC_EnableIRQ(TIM2_IRQn);
}

MotionFX_manager_run(&data_in,&
data_out,MFX_DELTATIME);
snprintf(data, 128, "%.3f;%.3f;%.3f
|.3f;|.3f;|.3f|.3f;|.3f
;|.3f;|.3f;|.3f;|.3f\n",
data_in.acc[0], data_in.acc[1],
data_in.acc[2],
data_in.gyro[0],data_in.gyro[1],
data_in.gyro[2],
data_in.mag[0],data_in.mag[1],data_in
.mag[2],
data_out.rotation_9X[0],data_out.
rotation_9X[1],data_out.
rotation_9X[2]);
}

//transmit data
CDC_Transmit_FS((uint8_t *)data,strlen(data
));
flag=0;

HAL_NVIC_EnableIRQ(TIM3_IRQn);
}
```

}

### A.1.7 Script Test Computation Mode per il modulo APP

Di seguito si riporta il codice utilizzato per implementare la funzione di Test Computation Mode all'interno del modulo APP.

---

```
%-----Init-----%
f=figure;
.
.
.
g=figure;
.
.
.
h=figure
.
.
.

%Sensor fusion algorithm init
T=0.01;
qk=[0 0 0 0] .';
p_qk=0;
Pk=eye(4);
p_Pk=eye(4);
data=strings(1,10000);
acc=zeros(6000,3);
gyro=zeros(6000,3);
rotation=zeros(6000,3);
%set 0 for 6DOF 1 otherwise
DOF=0;
```

```
count=length(Data);  
  
%-----Read-----%  
for c=1:count  
  
[accData,gyroData,~,rotationData]=read(Data(c),DOF);  
  
%Acceleration on x-axe  
acc(c,1)=accData(1);  
  
%Acceleration on y-axe  
acc(c,2)=accData(2);  
  
%Acceleration on z-axe  
acc(c,3)=accData(3);  
  
%Angular velocity on x-axe  
gyro(c,1)=gyroData(1);  
  
%Angular velocity on y-axe  
gyro(c,2)=gyroData(2);  
  
%Angular velocity on z-axe  
gyro(c,3)=gyroData(3);  
  
%Magnetic field on x-axe  
magneto(c,1)=gyroData(1);  
  
%Magnetic field on y-axe  
magneto(c,2)=gyroData(2);  
  
%Magnetic field on z-axe  
magneto(c,3)=gyroData(3);  
  
%Yaw  
rotation(c,1)=rotationData(1);
```

```
%Pitch
rotation(c, 2)=rotationData(2);

%Roll
rotation(c, 3)=rotationData(3);

end
%-----Sensor fusion-----
for c=1:count

%----Stage 0:Angular position estimation----
gyroRaw=[gyro(c,1),gyro(c,2),gyro(c,3)];
[qk,Pk] = Stage0(gyroRaw,p_qk,p_Pk,T);

%-----Stage 1:correction position-----
accRaw=[acc(c,1),acc(c,2),acc(c,3)];
[qk1, Pk1]= Stage1(accRaw,qk,Pk);

%-----Stage 2:correction positio-----
if (DOF==1)
magnetoRaw=[magneto(c,1),magneto(c,2),magneto(c,3)];
[qk2, Pk2]= Stage2(magnetoRaw,qk1,Pk1);
[yaw,pitch,roll]=quat2angle(qk1');

%Feedbacks loop for 9DOF
p_qk=qk2;
p_Pk=Pk2;
end
else
%Feedbacks loop for 6DOF
p_qk=qk1;
p_Pk=Pk1;
end
%-----Plot stime-----
yaw=yaw*180/pi;
```

```
pitch=pitch*180/pi;  
roll=roll*180/pi;  
  
addpoints(rollLine,c*T,roll);  
addpoints(rollSTMLine,c*T,rotation(c,3));  
  
addpoints(pitchLine,c*T,pitch);  
addpoints(pitchSTMLine,c*T,rotation(c,2));  
  
addpoints(yawLine,c*T,yaw);  
addpoints(yawSTMLine,c*T,rotation(c,1));  
  
addpoints(angularX,c*T,gyro(c,1)*180/pi);  
addpoints(angularY,c*T,gyro(c,2)*180/pi);  
addpoints(angularZ,c*T,gyro(c,3)*180/pi);  
end
```

## A.2 Testing

### A.2.1 Script per la stima del tempo di lettura di dati provenienti dall’IMU

Di seguito si riporta il codice utilizzato per stimare il tempo di lettura di dati provenienti dall’unità di misura inerziale utilizzando un canale I2C in fast-mode.

---

```
//Test I2C speed  
while(1) {  
    if(id<1000) {  
  
        if(!I2C_ReadData(LSM9DS1_I2C_BADD,  
                           LSM9DS1_WHO_AM_I,&result,1)) {  
            if( result == LSM9DS1_WHO_AM_I_VALUE)  
                id++;  
        }  
        else
```

```

{
    lost++;
}

if(id==1000) {
    t1=HAL_GetTick()-t0;
    HAL_Delay(2);
    snprintf(data, 10, "%Lu",t1);
    CDC_Transmit_FS((uint8_t *)data,strlen(data));
}
}
}

```

## A.2.2 Script per la stima del tempo di trasmissione di dati al modulo App

Di seguito si riporta il codice utilizzato per stimare il tempo di trasmissione sul canale USB utilizzato nella classe CDC.

---

```

//Test usb CDC speed
while(1)
{
    snprintf(data, 10, "%d|%d\n",id,lost);
    if(id<1000) {
        if(CDC_Transmit_FS((uint8_t *)data,strlen(
            data))== USBD_OK)
        {
            id++;
        }
        else{
            lost++;
        }

    if(id==1000) {
        t1=HAL_GetTick()-t0;

```

```
    snprintf(data, 10, "%Lu\n", t1);
    CDC_Transmit_FS((uint8_t *)data, strlen(data
        ));
}

}
```

### A.2.3 Script per il sistema ausiliario Arduino per l'analisi qualitativa

Di seguito si riporta il codice utilizzato su Arduino per muovere il servo motore al fine di effettuare l'analisi qualitativa degli algoritmi di *Sensor Fusion*.

---

```
#include <Servo.h>
Servo myservo; // create servo object to control a
                servo
// a maximum of eight servo objects can be created
int pos = 0; // variable to store the servo position
int buttonState=0; // variable for reading the
                    pushbutton status
int count=0;

// constants won't change. They're used here to set pin
// numbers:
const int buttonPin = 2; // the number of the
                        pushbutton pin
const int ledPin = 13; // the number of the LED pin

void setup() {
    // attaches the servo on pin 9 to the servo
    // object
    myservo.attach(9);
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
```

```
pinMode(buttonPin, INPUT);
myservo.write(0);
}

void loop() {
    while(digitalRead(buttonPin) != HIGH) {
        digitalWrite(ledPin, LOW);
    };

    for (int i=0; i<2; i++) {
        for(int j=0; j<5; i++) {
            digitalWrite(ledPin, HIGH);
            myservo.write(90);
            delay(500);
            myservo.write(0);
            delay(500);
        }
        delay(1000);
    }
}
```

# Appendice B

## Filtro di Kalman

In questa sezione si riporta l'appendice riguardante la teoria e i fondamenti del filtro di Kalman [24].

Il filtro di Kalman può essere utilizzato per stimare lo stato passato, presente e futuro di un sistema nel quale le variabili di input e output non sono completamente note in quanto affette da rumore.

Il filtro permette di ottenere una prima *stima a priori*. Nell'ambito di questa tesi lo stato è l'assetto e la stima è fatta a partire dai dati del giroscopio. Successivamente la stima a *priori* viene raffinata usando l'accelerometro ed eventualmente anche il magnetometro, per ottenere quella che viene chiamata *stima a posteriori*. Come è facile capire, è possibile usare il filtro di Kalman in tutti i casi in cui è necessario fondere dati provenienti da sorgenti diverse, ottenendo una stima del sistema migliore col crescere dei sensori utilizzati, che sono tuttavia affetti da rumore.

### B.1 Descrizione teorica del filtro di Kalman

E' possibile descrivere un sistema a tempo discreto con un'equazione stocastica lineare del tipo:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (\text{B.1})$$

Dove  $x_k \in \mathbb{R}^n$  è il vettore che descrive lo stato del sistema,  $u_k$  è il vettore di dimensione  $l$  con le variabili d'ingresso del sistema,  $A$  è una matrice  $n \times n$  che descrive l'evoluzione del sistema senza input,  $B$  è la matrice  $n \times l$  che lega l'evoluzione del sistema con il vettore d'ingressi  $u_k$  ed infine  $w_{k-1}$  è il vettore contenente i rumori del processo.

Le variabili di output sono descritte invece dalla seguente equazione:

$$z_k = Hx_k + v_k \quad (\text{B.2})$$

Dove  $z_k$  è il vettore con le variabili di output,  $H$  è la matrice che lega lo stato del sistema con l'output e  $v_k$  è il vettore che descrive la misura del rumore. Si deve assumere che il processo e la misura del rumore siano indipendenti, bianchi e con una distribuzione Gaussiana:

$$p(w) \sim N(0, Q) \quad (\text{B.3})$$

$$p(v) \sim N(0, R) \quad (\text{B.4})$$

Le due distribuzioni hanno una media pari a zero e una covarianza descritta dalla matrice  $Q$  e  $R$ . Generalmente, le matrici  $Q$  e  $R$  possono variare ad ogni iterazione del filtro, ma per semplicità nell'ambito di questa tesi sono considerate costanti.

Si definisce  $\hat{x}_k^-$  la stima *a priori* dello stato del sistema, mentre con  $\hat{x}_k$  la stima *a posteriori*. Se  $x_k$  è lo stato reale del sistema, è possibile definire l'errore *a priori* e *a posteriori* della stima del sistema:

$$e_k^- \equiv x_k - \hat{x}_k^- \quad (\text{B.5})$$

$$e_k \equiv x_k - \hat{x}_k \quad (\text{B.6})$$

Poiché questo sistema è caratterizzato da un rumore descritto tramite variabili statistiche, è possibile caratterizzarla statisticamente anche l'errore della stima dello stato del sistema. In particolare, la covarianza dell'errore nella stima *a priori* e *a posteriori* è rispettivamente:

$$P_k^- = E[e_k^- e_k^{-T}] \quad (\text{B.7})$$

$$P_k = E[e_k e_k^T] \quad (\text{B.8})$$

La stima *a posteriori* dello stato del sistema è data dalla seguente equazione:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \quad (\text{B.9})$$

Dove si lega la stima *a priori* con quella che viene chiamata misura innovativa o residuo. Questa è la differenza tra l'attuale misura  $z_k$  e la misura predetta tramite

$$H\hat{x}_k^-.$$

Da B.9 si può notare che la misura a *priori*  $\hat{x}_k^-$  è corretta con una quantità data dalla moltiplicazione del residuo con la matrice  $K$  di dimensione  $n \times m$ . Questa matrice è detta **matrice del guadagno di Kalman** ed è calcolata per minimizzare  $P_k^-$ , la matrice di covarianza dell'errore a *priori*, al fine di ottenere la stima ottima dello stato del sistema.

Sostituendo B.9 alla definizione di  $e_k$  e in B.8, derivando rispetto a  $K$  e risolvendo per trovare il minimo, si può scrivere la seguente equazione per calcolare la matrice del guadagno di Kalman:

$$K_k = P_k^{-1} H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^{-1} H^T}{H P_k^- H^T + R} \quad (\text{B.10})$$

Si noti come anche la matrice  $K$  potrebbe cambiare ad ogni iterazione del filtro. Si possono fare delle considerazioni interessanti per capire meglio il filtro di Kalman. Infatti, se la covarianza dell'errore  $R$  è vicina allo zero, il filtro pesa più la misura  $z_k$  rispetto alla stima a *priori*  $\hat{x}_k^-$  per ricostruire lo stato del sistema. In questo caso:

$$\lim_{R \rightarrow 0} K_k = H^{-1}; \lim_{R \rightarrow 0} \hat{x}_k = H^{-1} z_k \quad (\text{B.11})$$

Nel caso opposto che la matrice di covarianza dell'errore della stima a *priori* sia vicina allo zero, la matrice  $K$  tende anch'essa a zero. In questo caso, il residuo conta meno per la stima dello stato a *posteriori*, che tende ad essere uguale alla stima a *priori*:

$$\lim_{P_k^- \rightarrow 0} K_k = 0; \lim_{P_k^- \rightarrow 0} \hat{x}_k = \hat{x}_k^- \quad (\text{B.12})$$

Si può concludere che, se il processo e la misura del rumore sono Gaussiani, bianchi e indipendenti, allora la stima del sistema  $x_k$  è una variabile Gaussiana con media pari a  $\hat{x}_k$  e covarianza  $P_k$ .

## B.2 L'algoritmo del filtro di Kalman discreto

Come si è appena visto, il filtro di Kalman prova a stimare lo stato di un sistema con qualcosa di simile ad un feedback di controllo. L'equazioni del filtro sono divise in due differenti tipi: l'**equazioni di update** dello stato, anche chiamate equazioni di predizione, che aggiornano la sitma a *priori* dello stato del sistema e l'**equazioni di misura** che sono responsabili di correggere il feedback e la stima a *posteriori*.

L'equazioni di predizione sono:

$$x_k^- = A\hat{x}_{k-1} + Bu_k \quad (\text{B.13})$$

$$P_k^- = AP_{k-1}A^T + Q \quad (\text{B.14})$$

Si può notare che la stima a *priori* di  $x_k^-$  è fatta a partire dalla stima a posteriori dello stato del sistema ottenuta nell'iterazione precedente  $k - 1$ .

La matrice di covarianza dell'errore dello stato, stimato a *priori*, è anch'essa calcolata a partire da quella calcolata a posteriori nell'iterazione precedente e poi sommata con la matrice  $Q$ , che è la covarianza del rumore del processo. L'equazioni di misura sono:

$$K_k = P_k^- H^T (HP_k^- HT + R)^{-1} \quad (\text{B.15})$$

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \quad (\text{B.16})$$

$$P_k = (I - K_k H)P_k^- \quad (\text{B.17})$$

Nel primo step la matrice del guadagno di Kalman  $K_k$  è aggiornata. Questo è necessario per misurare l'output del sistema  $z_k$  e per calcolare la stima a *posteriori*  $\hat{x}_k$ . Nell'ultimo step, si calcola, a *posteriori*, la matrice di covarianza dell'errore  $P_k$ . Per ogni iterazione del filtro, l'equazione B.13 e B.14 sono utilizzate per calcolare la stima a *priori* dello stato allo step  $k$ , quindi si utilizzano B.15, B.16 e B.17 per calcolare la stima a *posteriori*. Allo step successivo  $k + 1$  viene ripetuto quanto appena detto utilizzando la stima a posteriori calcolata nell'iterazione precedente, come mostrato nella figura seguente:

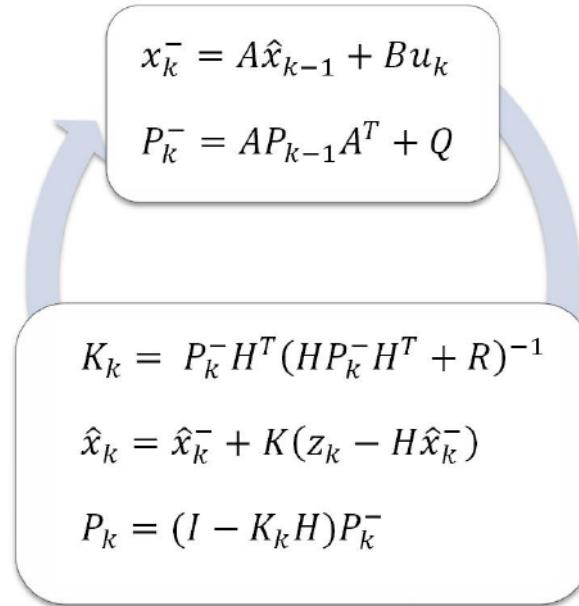


Figura B.1: Rappresentazione dell'algoritmo di Kalman per un sistema tempo discreto

E' ora necessario vedere come inizializzare il filtro e quali valori bisogna stimare e misurare. Per prima cosa, il sistema deve essere modellato, quindi è necessario stimare le matrici  $A$  e  $B$  che descrivo l'evoluzione del sistema e la matrice  $H$  che lega lo stato del sistema con la misura dell'output. E' necessario inoltre misurare o calcolare la matrice  $R$  che, come detto, rappresenta la covarianza dell'errore delle misure.

La matrice  $Q$  non è sempre direttamente misurabile, questo perché lo stato del sistema di solito non può essere misurato.

All'inizio, il filtro deve essere inizializzato con qualche valore di  $\hat{x}_0$  e  $P_0$ . Questi valori non sono veramente importanti in quanto il filtro è ben implementato, dopo un certo numero di iterazioni lo stato del sistema convergerà alla stima corretta indipendentemente dal valore  $\hat{x}_0$  scelto. Tuttavia una scelta corretta va ad inficiare sulla lunghezza e sull'intensità di questo transitorio iniziale.

Una considerazione finale: se la matrice  $Q$  e  $R$  sono costanti, dopo la prima iterazione del filtro anche i valori  $P_k$  e  $K_k$  si stabilizzeranno a valori costanti. In questo caso, questi valori possono essere precalcolati durante la fase di settaggio e memorizzati.

### B.3 Il filtro di Kalman esteso

Si è visto precedentemente, come funziona il filtro di Kalman e come sia possibile implementarlo a partire da equazioni differenziali lineari stocastiche che descrivo l’evoluzione del sistema. Il problema è che tutte le considerazioni fatte valgono solo per i sistemi lineari, ma la maggior parte delle applicazioni reali riguardano sistemi non lineari e il lavoro di questa tesi non fa eccezione.

Si deve considerare una nuova versione del filtro di Kalman, il **filtro di Kalman esteso** (EKF). Questa versione funziona con sistemi non lineare e usa di linearizzarla attorno al punto di lavoro attraverso una matrice Jacobiana.

In questo caso il sistema è descritto da un’equazione stocastica non lineare:

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \quad (\text{B.18})$$

$$z_k = h(x_k, v_k) \quad (\text{B.19})$$

I termini hanno gli stessi significati della versione lineare del filtro B.1 con la differenza che ora appaiono all’interno di un’equazione non lineare.

Deve essere fatto presente che, sicché si tratta di un’equazione non lineare e nonostante  $w_k$  e  $v_k$  siano rumori bianchi Gaussiani, il rumore generato non è Gaussiano. Tuttavia grazie alla linearizzazione intorno al punto di lavoro, questo può essere considerato come tale.

Di solito, è possibile approssimare lo stato e le misure considerando il rumore uguale a zero:

$$\tilde{x}_k = f(x_{k-1}, u_k, 0) \quad (\text{B.20})$$

$$\tilde{z}_k = h(\tilde{x}_k, 0) \quad (\text{B.21})$$

Per stimare il sistema descritto da un’equazione non lineare è possibile scrivere due equazioni linearizzate che includono la stima di  $\tilde{x}_k$  e  $\tilde{z}_k$  viste precedentemente:

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1} \quad (\text{B.22})$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k \quad (\text{B.23})$$

Nelle quali,  $x_k$  e  $z_k$  sono rispettivamente lo stato del sistema e il vettore delle misure,  $\tilde{x}_k$  e  $\tilde{z}_k$  sono lo stato del sistema e il vettore delle misure approssimate da

B.20 e B.21,  $\hat{x}_k$  è la stima a *posteriori* dello stato all’iterazione  $k$ ,  $w_k$  e  $v_k$  sono i rumori del processo e delle misure.

La matrice Jacobiana  $A$  introdotta è la matrice delle derivate parziali di  $f$  rispetto a  $x$ :

$$A_{[i,j]} = \frac{\partial f_i}{\partial x_j}(\hat{x}_k, u_k, 0) \quad (\text{B.24})$$

La matrice  $W$  è la matrice Jacobiana delle derivate parziali di  $f$  rispetto a  $w$ :

$$W_{[i,j]} = \frac{\partial f_i}{\partial w_j}(\hat{x}_k, u_k, 0) \quad (\text{B.25})$$

La matrice  $H$  è la matrice Jacobiana delle derivate parziali di  $h$  rispetto ad  $x$ :

$$H_{[i,j]} = \frac{\partial h_i}{\partial x_j}(\tilde{x}_k, 0) \quad (\text{B.26})$$

La matrice  $V$  è la matrice Jacobiana delle derivate parziali di  $h$  rispetto a  $v$ :

$$V_{[i,j]} = \frac{\partial h_i}{\partial v_j}(\tilde{x}_k, 0) \quad (\text{B.27})$$

Ancora una volta si deve sottolineare che le matrici sopracitate potrebbero cambiare ad ogni iterazione del filtro. A questo punto si applicano le notazioni introdotte per l’errore durante la stima dello stato e delle misure approssimate:

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k \quad (\text{B.28})$$

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k \quad (\text{B.29})$$

Le precedenti equazioni possono essere approssimate nel seguente modo:

$$\tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \epsilon_k \quad (\text{B.30})$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k \quad (\text{B.31})$$

Ora due nuove variabili casuali e indipendenti sono state introdotte. Queste hanno media pari a zero e la covarianza dell’errore è data dalle matrice  $WQW^T$  e  $VRV^T$ , con  $Q$  e  $R$  matrici di covarianza dell’errore del processo e della misura rispettivamente.

Queste equazioni derivate sono lineari e simili a quelle che descrivono l’evoluzione del sistema lineare. Quindi si possono applicare ad un secondo ipotetico filtro di

Kalman per stimare l'errore  $\tilde{e}_{x_k}$ . Questa stima può a sua volta essere usata con B.28 per ottenere la stima a *posteriori* dello stato del sistema non lineare:

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k \quad (\text{B.32})$$

La variabile casuale  $\tilde{e}_{x_k}$  ha una distribuzione Gaussiana con una media pari a zero e una covarianza dell'errore  $E(\tilde{e}_{x_k} \tilde{e}_{x_k}^T)$ . Così, se il valore predetto di  $\hat{e}_k$  è zero, l'equazione usata per stimarlo si riduce a:

$$\hat{e}_k = K_k \tilde{e}_{z_k} \quad (\text{B.33})$$

Sostituendo questa relazione in B.32 e poi in B.29 si ottiene:

$$\hat{x}_k = \tilde{x}_k + K_k \tilde{e}_{z_k} = \tilde{x}_k + K_k(z_k - \tilde{z}_k) \quad (\text{B.34})$$

Da questo risultato si vede che in realtà non è necessario usare un secondo filtro di Kalman perché l'equazione dipende solo da  $\tilde{x}_k$  e  $\tilde{z}_k$ , che sono definite in B.20 e B.21, mentre  $K_k$  è il guadagno del filtro di Kalman definito in B.15.

In conclusione, l'equazione del filtro di Kalman esteso sono:

$$\tilde{x}_k = x_k^- = f(\hat{x}_{k-1}, u_k, 0) \quad (\text{B.35})$$

$$P_k^- = A_k P_{k-1}^- A_k^T + W_k Q_{k-1}^- W_k^T \quad (\text{B.36})$$

Mentre l'equazioni di correzione sono:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \quad (\text{B.37})$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0)) \quad (\text{B.38})$$

$$P_k = (I - K_k H_k) P_k^- \quad (\text{B.39})$$

In queste equazioni finali, al fine di essere coerenti con il filtro di Kalman la notazione  $\tilde{x}_k$  è stato sostituita da  $x_k^-$ .

Si noti come le operazioni del filtro di Kalman esteso non siano cambiate rispetto al filtro "normale", con equazioni di predizione e correzione simili eseguite ad ogni iterazione. La grande differenza è che ora le matrici  $A$  e  $H$  più le matrici  $W$  e  $V$  sono matrici Jacobiane con derivate parziali delle funzioni non lineari  $f$  e  $h$ . Di solito, queste matrici sono ricalcolate ad ogni iterazione  $k$ , come mostrato in figura:

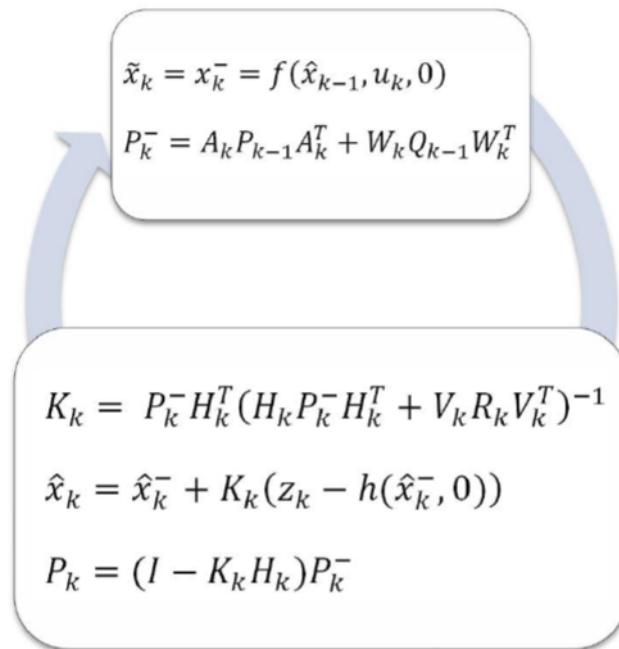


Figura B.2: Rappresentazione dell'algoritmo di Kalman esteso per un sistema tempo discreto non lineare

# Bibliografia

- [1] Towards a Better Understanding of Context and Context-Awareness <ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf>
- [2] Sistema di posizionamento globale [https://it.wikipedia.org/wiki/Sistema\\_di\\_posizionamento\\_globale#cite\\_note-1](https://it.wikipedia.org/wiki/Sistema_di_posizionamento_globale#cite_note-1)
- [3] Indoor positioning system [https://en.wikipedia.org/wiki/Indoor\\_positioning\\_system](https://en.wikipedia.org/wiki/Indoor_positioning_system)
- [4] Progettazione e realizzazione di un Indoor Positioning System basato su geomagnetismo e sensor fusion [http://amslaurea.unibo.it/12840/1/federico\\_torsello\\_tesi.pdf](http://amslaurea.unibo.it/12840/1/federico_torsello_tesi.pdf)
- [5] Luca Pappalardo, "Localizzazione - Problema, Tecniche, Algoritmi - Reti mobili: Ad Hoc e di sensori", 2011, <http://didawiki.di.unipi.it/lib/exe/fetch.php/rhs/localizzazione.pdf>
- [6] Cuccado, De Franceschi, Fauri, Sartor, "Analisi di algoritmi di autolocalizzazione per reti di sensori wireless", 2007, <https://art.torvergata.it/retrieve/handle/2108/773/6945/Paolo-Sperandio-Tesi-PhD.pdf>
- [7] An adaptive indoor positioning system based on Bluetooth Low Energy RSSI <https://www.politesi.polimi.it/bitstream/10589/92284/3/NicolaCinefra770910TesiDefinitiva.pdf>
- [8] Harrop P, Raghu D. Mobile Phone Indoor Positioning Systems (IPS) and Real Time Locating Systems (RTLS) 2014-2024. Forecasts, Players, Opportunities. IDTechEx

- [9] Ugur Bekcibasi, "Increasing RSSI Localization Accuracy with Distance Reference Anchor in Wireless Sensor Networks", 2014, [http://www.uni-obuda.hu/journal/Bekcibasi\\_Tenruh\\_54.pdf](http://www.uni-obuda.hu/journal/Bekcibasi_Tenruh_54.pdf)
- [10] Mak LC, Furukawa T. A ToA-based Approach to NLOS Localization Usiong Low-Frequency Sound. ACRA2006 (Auckland, New Zealand); 2006
- [11] A Simple Technique for angle of arrival measurement [https://www.researchgate.net/publication/4368716\\_A\\_Simple\\_Technique\\_for\\_angle\\_of\\_arrival\\_measurement](https://www.researchgate.net/publication/4368716_A_Simple_Technique_for_angle_of_arrival_measurement)
- [12] Generalized Geometric Triangulation Algorithm for Mobile Robot Absolute Self-Localization <https://pdfs.semanticscholar.org/dee6/fb124433cac10744af9502b165ffdec202.pdf>
- [13] Sensor fusion [https://en.wikipedia.org/wiki/Sensor\\_fusion](https://en.wikipedia.org/wiki/Sensor_fusion)
- [14] Jeroen Hol "Sensor Fusion and Calibration of Inertial Sensors, Vision, Ultra-Wideband and GPS" <https://www.xsens.com/wp-content/uploads/2014/pdf/Hol2011%20-%20Dissertation.pdf>
- [15] Janusz Bryzek "Principles of MEMS" <https://www.wiley.com/legacy/wileychi/hbmsd/pdfs/mm573.pdf>
- [16] MEMS Accelerometer <http://www.instrumentationtoday.com/mems-accelerometer/2011/08/>
- [17] H. Titterton and J. L. Weston. Strapdown inertial navigation technology. IEE radar, sonar, navigation and avionics series. Peter Peregrinus Ltd., Stevenage, UK, 1997. ISBN 0863413587.
- [18] Forza di Coriolis, Wikipedia [https://it.wikipedia.org/wiki/Forza\\_di\\_Coriolis](https://it.wikipedia.org/wiki/Forza_di_Coriolis)
- [19] The Development of Micromachined Gyroscope Structure and Circuitry Technology, Dunzhu Xia, Cheng Yu and Lun Kong
- [20] TECNICHE DI STIMA E CONTROLLO DI ASSETTO DI SATELLITI. Andrea Fagiani. <http://control.disp.uniroma2.it/carnevale/archivio/Tesi/andreafagianiM/tesi.pdf>

- [21] DETERMINAZIONE D'ASSETTO MEDIANTE IL FILTRO DI KALMAN ESTESO MOLTIPLICATIVO. Fabio Cisaria.
- [22] Aircraft principal axes, Wikipedia. [https://en.wikipedia.org/wiki/Aircraft\\_principal\\_axes](https://en.wikipedia.org/wiki/Aircraft_principal_axes)
- [23] Gimbal lock, Wikipedia. [https://en.wikipedia.org/wiki/Gimbal\\_lock](https://en.wikipedia.org/wiki/Gimbal_lock)
- [24] Advanced algorithms and architectures for MEMS inertial sensor platforms in orientation tracking and in fall detection applications. Simone Sabatelli. Università di Pisa.
- [25] MotionFX middleware library in X-CUBE-MEMS1 software expansion for STM32Cube.  
[http://www.st.com/content/ccc/resource/technical/document/user\\_manual/group0/31/0e/66/39/cb/f7/4e/cd/DM00394369/files/DM00394369.pdf/jcr:content/translations/en.DM00394369.pdf](http://www.st.com/content/ccc/resource/technical/document/user_manual/group0/31/0e/66/39/cb/f7/4e/cd/DM00394369/files/DM00394369.pdf/jcr:content/translations/en.DM00394369.pdf)
- [26] Filtro di Kalman, Wikipedia.  
[https://it.wikipedia.org/wiki/Filtro\\_di\\_Kalman](https://it.wikipedia.org/wiki/Filtro_di_Kalman)
- [27] I2C, Wikipedia.  
<https://it.wikipedia.org/wiki/I%C2%B2C>
- [28] Description of STM32F4 HAL and LL drivers.  
[http://www.st.com/content/ccc/resource/technical/document/user\\_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf](http://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf)
- [29] Universal Serial Bus Class Definitions for Communication Devices.  
[https://cscott.net/usb\\_dev/data/devclass/usbccdc11.pdf](https://cscott.net/usb_dev/data/devclass/usbccdc11.pdf)
- [30] CDC Class Overview.  
<https://doc.micrium.com/display/DOC/CDC+Class+Overview>